```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
```

```
DDDDDDDD    DDDDDDDD      CCCCCCCC  MM      MM  PPPPPPPP
DDDDDDDD    DDDDDDDD      CCCCCCCC  MM      MM  PPPPPPPP
DD      DD  DD      DD  CC          MMMM  MMMM  PP      PP
DD      DD  DD      DD  CC          MMMM  MMMM  PP      PP
DD      DD  DD      DD  CC          MM  MM  MM  PP      PP
DD      DD  DD      DD  CC          MM  MM  MM  PP      PP
DD      DD  DD      DD  CC          MM      MM  PPPPPPPP
DD      DD  DD      DD  CC          MM      MM  PPPPPPPP
DD      DD  DD      DD  CC          MM      MM  PP
DD      DD  DD      DD  CC          MM      MM  PP
DD      DD  DD      DD  CC          MM      MM  PP              ....
DD      DD  DD      DD  CC          MM      MM  PP              ....
DDDDDDDD    DDDDDDDD      CCCCCCCC  MM      MM  PP              ....
DDDDDDDD    DDDDDDDD      CCCCCCCC  MM      MM  PP              ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

(3)	178	DECLARATIONS
(19)	754	Command Interrupter
(24)	921	Common event dispatcher
(26)	1052	Error
(28)	1079	Command error handling routines
(32)	1145	Set charateristics
(34)	1425	Initate start of protocol
(36)	1482	Reset variables
(37)	1572	Start acknowledge
(39)	1617	Start maintenance mode
(40)	1640	Receive message
(42)	1868	Transmit message
(44)	2031	Check the receives message header
(46)	2088	Reply timing
(47)	2117	Queue empty - protocol may have work
(49)	2148	Selection timeout clock
(56)	2269	Queue and requeue the message to be transmitted
(60)	2544	Calculate the CRC
(62)	2574	Setup headers for maint and data messages
(63)	2632	Set up control message headers
(65)	2741	Real clock timer
(67)	2948	Reply timeout clock queues and timer handling
(68)	3057	Halt
(69)	3124	Maintenance mode
(70)	3191	Return error counters
(72)	3235	Routines to handle sending and logging NAK's
(78)	3412	Routines to increment threshold counter
(80)	3465	Read or clear error counters

```
0000     1
0000     2          .TITLE DDCMP - Software DDCMP
0000     3          .IDENT 'V04-000'
0000     4
0000     5  ;
0000     6  ;*****************************************************************************
0000     7  ;*                                                                           *
0000     8  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     9  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
0000    10  ;*  ALL RIGHTS RESERVED.                                                     *
0000    11  ;*                                                                           *
0000    12  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    13  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    14  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    15  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    16  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    17  ;*  TRANSFERRED.                                                             *
0000    18  ;*                                                                           *
0000    19  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    20  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    21  ;*  CORPORATION.                                                             *
0000    22  ;*                                                                           *
0000    23  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    24  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
0000    25  ;*                                                                           *
0000    26  ;*                                                                           *
0000    27  ;*****************************************************************************
0000    28  ;
0000    29
0000    30  ;++
0000    31  ;
0000    32  ; FACILITY:     Software DDCMP
0000    33  ;
0000    34  ; ABSTRACT:
0000    35  ;
0000    36  ; This is an implementation of DDCMP in software. It performs a subset
0000    37  ; of the functions described in the DDCMP specification version 4.0.
0000    38  ; It is called as a subroutine from the sync line driver.
0000    39  ;
0000    40  ; COMMON INPUTS:
0000    41  ;                R5 = Tributary fields block address
0000    42  ;                R6 = Command from the driver to DDCMP
0000    43  ;                R7 = Different codes associated with each command
0000    44  ;                R8 = Contains the address of the buffer to process
0000    45  ;                R9 = Contains the second data field
0000    46  ;
0000    47  ; COMMON OUTPUTS:
0000    48  ;                R6 = Command from DDCMP to driver
0000    49  ;                R7 = Associated codes
0000    50  ;                R8 = Address of the requested buffer
0000    51  ;                R9 = Second data field
0000    52  ;
0000    53  ;       All other registers are destroyed.
0000    54  ;
0000    55  ;
0000    56  ; ENVIRONMENT:
0000    57  ;
```

```
0000     58 ;       I/O driver, Kernel mode, driver fork IPL.
0000     59 ;
0000     60 ;--
0000     61 ;
0000     62 ; AUTHOR: Meg Dumont,           CREATION DATE: 1-Apr-1981
0000     63 ;
0000     64 ; MODIFIED BY:
0000     65 ;
0000     66 ;       V03-018 MMD0312         Meg Dumont,     19-Jul-1984  9:39
0000     67 ;               Fix the CRCTABLE it wasn't generating the right CRCs'.
0000     68 ;               Also a fix to the NAK logging routine and a fix to the
0000     69 ;               RCVMSG routine.
0000     70 ;
0000     71 ;       V03-017 MMD0293         Meg Dumont,     23-Apr-1984  15:28
0000     72 ;               Fix the handling of retransmit messages. There is a race
0000     73 ;               condition that currently exists in the DDCMP interface
0000     74 ;               where is a XMT message is in the drivers XMT_INPR
0000     75 ;               or POSTQ then retransmission of messages can happen
0000     76 ;               out of order. To avoid this the protocol will check
0000     77 ;               all places that the driver may keep a xmt message after
0000     78 ;               it is xmt. If the protocol finds XMT message in the
0000     79 ;               queue then it will take it from the device and put
0000     80 ;               it back on the RTOQ. This is really a hack and the
0000     81 ;               hope is that some future version of the protocol will
0000     82 ;               handle the XMT/RTO situtation better. Incidentally all
0000     83 ;               of this is done at DIPL inorder to synchronize the
0000     84 ;               various queues.
0000     85 ;
0000     86 ;       V03-016 MMD0261         Meg Dumont,     22-Mar-1984  14:38
0000     87 ;               Fix so that Maintenance mode isn't entered until message
0000     88 ;               passes the CRC checks.
0000     89 ;
0000     90 ;       V03-015 MMD0223         Meg Dumont,     11-Jan-1984  10:15
0000     91 ;               Some fixes to the new error incrementing routine. And
0000     92 ;               all CASE statements execpt the one in DDCMP$SETCHAR
0000     93 ;               became $DISPATCH statements.
0000     94 ;
0000     95 ;       V03-014 MMD0217         Meg Dumont,     6-Jan-1984  12:57
0000     96 ;               Fix to QUEUE_XMT which makes the setting of the bits
0000     97 ;               XMTQ$M_SELECT in XMTQ$B_FLAGS and TF$M_OWNSELT in
0000     98 ;               TF$B_SELTIM to be done at DEVICE IPL. This fixes a
0000     99 ;               bug where the select flag is send in a message
0000    100 ;               but the driver doesn't get informed and thus the
0000    101 ;               the XMTER doesn't get shut down nor does RTS get dropped.
0000    102 ;               This would show up (for example) as REMOTE PROCESS
0000    103 ;               errors on a multipoint control station.
0000    104 ;
0000    105 ;       V03-013 MMD0211         Meg Dumont,     9-Dec-1983  12:59
0000    106 ;               Fix to maintenance mode to ensure that DECnet shuts down
0000    107 ;               the line and brings it up in MOP mode. Fixed bug in
0000    108 ;               DDCMP$TIMER where in Trib mode an ACK could be sent even
0000    109 ;               if we did not own the select flag.
0000    110 ;
0000    111 ;       V03-012 MMD0192         Meg Dumont,     15-Jun-1983  13:19
0000    112 ;               Support for a new interface between the protocol timer and
0000    113 ;               the device driver timer routine, Fixes to the protocol
0000    114 ;               prompted by bugs found in the V3.5 protocol. These include
```

```
0000   115 ;       a problem in message retransmition where the protocol did
0000   116 ;       not retransmit messages in response to a NAK in half duplex
0000   117 ;       and trib mode. Cleaned up the handling of retransmitied
0000   118 ;       messages. Fixed a bug where a REP could be inccorrectly
0000   119 ;       sent after an NAK had been sent. Fixed a bug in the timer
0000   120 ;       handler where two error bits were being sent when a selection
0000   121 ;       timeout occured.
0000   122 ;
0000   123 ;       Other fixes include adding the capabilty to wait a longer
0000   124 ;       time before dropping the line to a peristant error on
0000   125 ;       startup. Also changing full duplex operation such that
0000   126 ;       it does not send idle ACKs on the line.
0000   127 ;
0000   128 ;   V03-011 MMD0190         Meg Dumont,     13-Jun-1983  11:28
0000   129 ;           Reset select timeout counter on receipt of a message with the
0000   130 ;           select flag. Fix to save the receive message size when calling
0000   131 ;           DDCMP$XMTMSG from DDCMP$RCVMSG. Added a check so that no
0000   132 ;           messages are queued to the XMTQ unless we own the select flag.
0000   133 ;
0000   134 ;   V03-010 0170            Meg Dumont,     3-May-1983  12:54
0000   135 ;           Add support for starting and stopping the DDCMP timer
0000   136 ;
0000   137 ;   V03-009 MMD0166         Meg Dumont,     27-Apr-1983  17:55
0000   138 ;           Cleanup of the routine DDCMP$XMTMSG
0000   139 ;
0000   140 ;   V03-008 MMD0008         Meg Dumont,     10-Oct-1982  20:27
0000   141 ;           Fix missing quote on .IDENT
0000   142 ;
0000   143 ;   V03-007 MMD0007         Meg Dumont,     8-Oct-1982  18:33
0000   144 ;           Add $DYNDEF to system definitions. Clean up handling of
0000   145 ;           select flags. Force sending of ACKs on half duplex point
0000   146 ;           to point selection timeouts.
0000   147 ;
0000   148 ;   V03-005 MMD0005         Meg Dumont,     30-Jun-1982  10:01
0000   149 ;           Add a special delta for tribs so that the user can
0000   150 ;           set via the selection timer in which interval the
0000   151 ;           station should deselect itself. Fixed a problem in
0000   152 ;           RCVMSG when rcving a STRT and in REPTIM took out
0000   153 ;           check for X > A so that it will always branch to check
0000   154 ;           entries on the RTOQ.
0000   155 ;
0000   156 ;   V03-004 MMD0004         Meg Dumont,     18-Jun-1982  17:04
0000   157 ;           Fix to always check for entries in the RTOQ. Change BNEQ
0000   158 ;           to BLSS when checking persitant errors. Add code so that
0000   159 ;           if no message is ever transmitted then device won't be
0000   160 ;           shutdown by max RTO's being reached.
0000   161 ;
0000   162 ;   V03-003 MMD0003         Meg Dumont,     25-May-1982  16:09
0000   163 ;           Fix to sync to IPL$_TIMER before queueing entries onto the
0000   164 ;           timer queue. Clean up of reply and selection timer handling.
0000   165 ;
0000   166 ;   V03-002 MMD0002         Meg Dumont,     23-Apr-1982  13:54
0000   167 ;           Change to send the select flag in message if no other message
0000   168 ;           to transmit. Also change when to check the max message
0000   169 ;           counter to be before sending the next message.
0000   170 ;
0000   171 ;   V03-001 MMD0001         Meg Dumont,     15-APR-1982     16:28
```

```
0000    172 ;
0000    173 ;
0000    174 ;**
0000    175
```

Bugfixes for multipoint Trib support

```
0000    177
0000    178          .SBTTL  DECLARATIONS
0000    179
0000    180 ;
0000    181 ; Include Files:
0000    182 ;
0000    183          $DLADEF
0000    184          $DLKDEF
0000    185          $DDCMPDEF
0000    186          $DYNDEF
0000    187          $GFDEF
0000    188          $IPLDEF                                 ; Interrupt prioty levels
0000    189          $MFDDEF
0000    190          $NMADEF
0000    191          $TFDEF
0000    192          $TQEDEF
0000    193          $XMTQDEF
0000    194
```

```
0000   196
0000   197 ;
0000   198 ; Macros
0000   199 ;
0000   200
0000   201 ;
0000   202 ; Bit definitions
0000   203 ;
0000   204 .MACRO  BITDEF  BLK,SYM,BITVAL
0000   205
0000   206         'BLK'$V_'SYM' = BITVAL
0000   207         'BLK'$M_'SYM' = 1@<BITVAL>
0000   208 .ENDM
0000   209
0000   210 ;
0000   211 ; Constant definitions
0000   212 ;
0000   213 .MACRO  CONSTDEF         BLK,SYM,CONSTVAL
0000   214
0000   215         'BLK'$C_'SYM' = CONSTVAL
0000   216
0000   217 .ENDM
0000   218
```

```
0000    220
0000    221 ;
0000    222 ; Define events
0000    223 ;
0000    224 .MACRO  DDCMPEVDEF      EV
0000    225
0000    226         .IF     B,'EV'
0000    227                 _$PHASE = _$PHASE + 1
0000    228         .IFF
0000    229                 _$EVCOD = _$EVCOD + 1
0000    230                 DDCMP$C_EV_'EV' = _$EVCOD
0000    231         .ENDC
0000    232 .ENDM
0000    233
0000    234 ;
0000    235 ; Calculate the size of the tables
0000    236 ;
0000    237 .MACRO  CALC     EV
0000    238
0000    239         .IF     NB,'EV'
0000    240                 DDCMP$C_MAX_EV = DDCMP$C_MAX_EV + 1
0000    241         .ENDC
0000    242 .ENDM
0000    243
0000    244
0000    245 ;
0000    246 ; Macro to build DDCMP tables
0000    247 ;
0000    248 .MACRO  DDCMPTABLES      VECTOR,?SS
0000    249
0000    250         DDCMP$C_MAX_EV  = -1            ; Total number of events
0000    251         _$EVCOD         = -1            ; Dummy variables
0000    252         _$PHASE         =  0
0000    253
0000    254         .IRP    A,<VECTOR>              ; Determine the table size
0000    255                 CALC A
0000    256         .ENDR
0000    257         _$PHASE         =  0
0000    258 SS:
0000    259         .IRP    A,<VECTOR>              ; Fill the tables
0000    260                 DDCMPEVDEF A
0000    261         .ENDR
0000    262 .=SS
0000    263 .ENDM
0000    264
```

```
0000   266
0000   267 ;
0000   268 ; Macro to initialize DDCMP state tables
0000   269 ;
0000   270
0000   271 .MACRO  STTAB   DFLT_STA,DFLT_ACT       ;Init the state transition table
0000   272
0000   273         DDCMP$C_NUMEVT  = DDCMP$C_MAX_EV + 1
0000   274         _$ACT_NDX       = 0             ; Init the action routine index
0000   275
0000   276         .IF     NDF ACT$_'DFLT_ACT'
0000   277                 ACT$_'DFLT_ACT' = _$ACT_NDX
0000   278                 _$ACT_NDX       = _$ACT_NDX + 1
0000   279         .ENDC
0000   280
0000   281 DDCMP$AB_STTAB: .REPT   DDCMP$C_NUMSTA * DDCMP$C_NUMEVT
0000   282         .BYTE   <DDCMP$C_STA_'DFLT_STA'@DDCMP$C_ACTBITS> + ACT$_'DFLT_ACT'
0000   283         .ENDR
0000   284
0000   285 DDCMP$ENDTABLES=.
0000   286
0000   287 .ENDM
0000   288
```

```
0000    290
0000    291 ;
0000    292 ; Macro to move the current position just past the table area
0000    293 ;
0000    294
0000    295 .MACRO  ENDSTTAB
0000    296         .=DDCMP$ENDTABLES
0000    297 .ENDM
0000    298
0000    299 ;
0000    300 ; Macro to move current position to the proper event area
0000    301 ;
0000    302
0000    303 .MACRO  EVENT EV                         ; Move PC to proper event
0000    304
0000    305         .=DDCMP$AB_STTAB + <EV * DDCMP$C_NUMSTA>
0000    306
0000    307 .ENDM
0000    308
0000    309 ;
0000    310 ; Macro to fill the build and enter the state transition table element
0000    311 ;
0000    312 .MACRO  STATE   CURSTA,NXSTA,ACTION,?LL ; Make table entry
0000    313
0000    314 LL:
0000    315         .=.+DDCMP$C_STA_'CURSTA'
0000    316         .IF     NDF ACT$_'ACTION'
0000    317                 ACT$_'ACTION'   = _$ACT_NDX
0000    318                 _$ACT_NDX       = _$ACT_NDX + 1
0000    319         .ENDC
0000    320
0000    321         .BYTE   <DDCMP$C_STA_'NXSTA' @DDCMP$C_ACTBITS> + ACT$_'ACTION'
0000    322
0000    323 .=LL
0000    324 .ENDM
0000    325
```

```
0000    327
0000    328 ;
0000    329 ; This macro translates into the CASEx instruction.  It calculates the
0000    330 ; "base" and "limit" parameters from the <index,displacement> list
0000    331 ; specfied in the 'vector' parameter.  The dispatch table is set up
0000    332 ; such that any unspecified index value within the bounds of the
0000    333 ; transfer vector is associated with a diplacement which transfers
0000    334 ; control to the first location after the CASE statement, i.e., behaves
0000    335 ; as if the index were out of bounds.
0000    336 ;
0000    337 ; Example:
0000    338 ;       $DISPATCH       R0,<-                   ; Message type in R0
0000    339 ;
0000    340 ;               ;index  displacement
0000    341 ;
0000    342 ;               <CI,    NSP$RCV_CI>,-           ; Process CI message
0000    343 ;               <CC,    NSP$RCV_CC>,-           ; Process CC message
0000    344 ;               <DI,    NSP$RCV_DI>,-           ; Process DI message
0000    345 ;               <DC,    NSP$RCV_DI>,-           ; Process DC message
0000    346 ;       >
0000    347 ;       BRW     NSP$RCV_ILLMSG                  ; Message type unknown
0000    348 ;
0000    349 .MACRO  $DISPATCH,      INDX,VECTOR,TYPE=W,NMODE=S^#,?MN,?MX,?S,?SS,?ZZ
0000    350 SS:
0000    351
0000    352         .MACRO  $DSP1,$DSP1_1
0000    353                 .IRP    $DSP1_2,$DSP1_1
0000    354                         $DSP2   $DSP1_2
0000    355                 .ENDR
0000    356         .ENDM
0000    357
0000    358         .MACRO  $DSP2,$DSP2_1,$DSP2_2
0000    359                 .=<$DSP2_1-MN>*2 + S
0000    360                 .WORD   $DSP2_2-S
0000    361         .ENDM
0000    362
0000    363
0000    364         .MACRO  $BND1,$BND1_1,$BND1_2,$BND1_3
0000    365                 $BND2   $BND1_1,$BND1_2
0000    366         .ENDM
0000    367
0000    368         .MACRO  $BND2,$BND2_1,$BND2_2
0000    369                 .IIF    $BND2_1,$BND2_2-.,      .=$BND2_2
0000    370         .ENDM
0000    371
0000    372         .MACRO  $BND    $BND_1,$BND_2
0000    373                 .IRP    $BND_3,<$BND_2>
0000    374                         $BNDT   $BND_1,$BND_3
0000    375                 .ENDR
0000    376         .ENDM
0000    377
0000    378         .=0
0000    379 ZZ:
0000    380         $BND    GT,<VECTOR>
0000    381 MX:
0000    382         $BND    LT,<VECTOR>
0000    383 MN:
```

```
0000    384          .=SS
0000    385
0000    386 CASE'TYPE        INDX,#<MN-ZZ>,NMODE'<MX-MN>
0000    387 S:
0000    388          .REPT   MX-MN+1
0000    389          .WORD   <MX-MN>*2 + 2
0000    390          .ENDR
0000    391
0000    392          .=S
0000    393
0000    394          $DSP1   <<VECTOR>>
0000    395
0000    396          .=<MX-MN>*2 + S + 2
0000    397
0000    398 .ENDM
0000    399
```

```
0000   401
0000   402 ; This macro sets up the network management definitions for the
0000   403 ; count fields
0000   404 ;
0000   405         .MACRO  COUNT   TYPE,BITMAP=NO,WIDTH=8,LOC
0000   406         $$$TYP = TYPE & NMA$M_CNT_TYP
0000   407         .IIF IDN <BITMAP><YES>, $$$TYP = $$$TYP!<NMA$M_CNT_MAP>
0000   408         $$$WID = 0                      ; Set reserved mask width
0000   409         .IIF IDN <WIDTH><8>, $$$WID = <1@NMA$V_CNT_WID>
0000   410         .IIF IDN <WIDTH><16>, $$$WID = <2@NMA$V_CNT_WID>
0000   411         .IIF IDN <WIDTH><32>, $$$WID = <3@NMA$V_CNT_WID>
0000   412         .IIF EQ $$$WID, .ERROR          ; Invalid bit width value
0000   413         MOVW    #<NMA$M_CNT_COU!$$$WID!$$$TYP>,LOC
0000   414         .ENDM   COUNT
0000   415
```

```
0000    417 ;
0000    418 ; DDCMP constants
0000    419 ;
0000    420
0000    421 ;
0000    422 ; Message fields with constant values
0000    423 ;
0000    424
0000    425 CONSTDEF        DDCMP,ACK,1             ; Acknowledge control message
0000    426 CONSTDEF        DDCMP,NAK,2             ; Negative acknowledge control message
0000    427 CONSTDEF        DDCMP,REP,3             ; Reply to msg number control message
0000    428 CONSTDEF        DDCMP,STRT,6            ; Start protocol control message
0000    429 CONSTDEF        DDCMP,STACK,7           ; Start acknowledged control message
0000    430
0000    431 CONSTDEF        DDCMP,ENQ,5             ; Control message identifier
0000    432 CONSTDEF        DDCMP,SOH,129           ; Data message identifier
0000    433 CONSTDEF        DDCMP,DLE,144           ; Maintenance message identifier
0000    434 CONSTDEF        DDCMP,SYN,150           ; SYN byte identifier
0000    435 CONSTDEF        DDCMP,DEL,255           ; DEL byte identifier
0000    436
0000    437 ;
0000    438 ; NAK reason codes
0000    439 ;
0000    440
0000    441 CONSTDEF        DDCMP,NAK1,1            ; Header CRC
0000    442 CONSTDEF        DDCMP,NAK2,2            ; Data CRC
0000    443 CONSTDEF        DDCMP,NAK3,3            ; REP response NUM field in REP # R
0000    444 CONSTDEF        DDCMP,NAK8,8            ; Buffer for incoming data unavailable
0000    445 CONSTDEF        DDCMP,NAK9,9            ; Receive overrun
0000    446 CONSTDEF        DDCMP,NAK16,16          ; RCV'd data message (COUNT) was too
0000    447                                         ; long for buffer
0000    448 CONSTDEF        DDCMP,NAK17,17          ; Message header format error
0000    449
0000    450 ;
0000    451 ; Misc constants
0000    452 ;
0000    453
0000    454 CONSTDEF        DDCMP,MSGCNT_LIMIT,7    ; Maximum msgs that one can send in
0000    455                                         ; a selection field the default
0000    456                                         ; for the MSGCNT field is normally 4
0000    457 CONSTDEF        DDCMP,MODULO,127        ; Mod for modulo arithmetic
0000    458
0000    459 ;
0000    460 ; Reply timer handling
0000    461 ;
0000    462
0000    463 CONSTDEF        DDCMP,REALCLOCK,1       ; Use a real time clock
0000    464 CONSTDEF        DDCMP,EOINTVL,2         ; Use end of selection interval
0000    465 CONSTDEF        DDCMP,NXTINTVL,3        ; Use next selection interval
0000    466 CONSTDEF        DDCMP,MILLI,1000        ; Set size of a millisecond
0000    467 CONSTDEF        DDCMP,QUA,250           ; Set size of a quater second
0000    468 CONSTDEF        DDCMP,RUNTQE,5*1000*1000 ; Delta for running protocol
0000    469                                         ; 1 tick / .5 seconds
0000    470 CONSTDEF        DDCMP,TRIBTQE,25*100*1000 ; Delta for protocol running in
0000    471                                         ; trib mode 1 tick / .25 seconds
0000    472 CONSTDEF        DDCMP,STPTQE,30*1000*1000 ; Delta for stopped protocol
0000    473                                         ; 1 tick / 3 seconds
```

```
0000    474
0000    475
0000    476 ;
0000    477 ; Threshold error constant fields
0000    478 ;
0000    479
0000    480 CONSTDEF        DDCMP,XMT_THRS,0        ; Start of the XMT threshold field
0000    481 CONSTDEF        DDCMP,RCV_THRS,3        ; Start of the RCV threshold field
0000    482 CONSTDEF        DDCMP,SEL_THRS,6        ; Start of the SEL threshold field
0000    483 CONSTDEF        DDCMP,THR_SIZE,3        ; Size of threshold field
0000    484 CONSTDEF        DDCMP,THR_MAX,7         ; Threshold maximum
0000    485
0000    486 ;
0000    487 ; Bit fields used
0000    488 ;
0000    489
0000    490 BITDEF          DDCMP,BIT8,8            ; Bit eight of the longword
0000    491 BITDEF          DDCMP,BIT16,16          ; Bit sixteen of the longword
0000    492 BITDEF          DDCMP,BIT24,24          ; Bit twenty-four of the longword
0000    493
```

```
0000    495
0000    496 ;
0000    497 ; STORAGE
0000    498 ;
0000    499 ; The following table lists the events that can occur in the protocol
0000    500 ; all input from the driver must map into these events.
0000    501 ;
0000    502
0000    503 DDCMPTABLES<-
0000    504         <RCV>,-                         ; Receive a data message or an ACK
0000    505         -                               ; NAK, or REP control message
0000    506         <STI>,-                         ; Receive a start initiate
0000    507         <STA>,-                         ; Receive a start acknowledge
0000    508         <RMM>,-                         ; Receive a maintenance message
0000    509         <XMT>,-                         ; Transmit a message
0000    510         <REP>,-                         ; Message to set up for reply timeout
0000    511         <SRT>,-                         ; User initiate start protocol
0000    512         <STP>,-                         ; User initiate stop protocol
0000    513         <QMT>,-                         ; User informs protocol that its XMTQ
0000    514         -                               ; is empty
0000    515         <MNT>,-                         ; User initiate set maintenance mode
0000    516         <ERR>,-                         ; User requests error buffers
0000    517         <CHR>,-                         ; Set line and station characteristics
0000    518         <LNK>,-                         ; The link was lost
0000    519         <LER>,-                         ; Error on msg RCV'd
0000    520         <STT>,-                         ; Start the DDCMP timer
0000    521         <SPT>,-                         ; Stop the DDCMP timer
0000    522         >
0000    523
0000    524 ;
0000    525 ; Define DDCMP protocol states
0000    526
00000000 0000   527                 DDCMP$C_STA_HLT = 0     ; DDCMP is halted
00000001 0000   528                 DDCMP$C_STA_RUN = 1     ; DDCMP is running
00000002 0000   529                 DDCMP$C_STA_IST = 2     ; Start initiated
00000003 0000   530                 DDCMP$C_STA_AST = 3     ; Start acknowledged
00000004 0000   531                 DDCMP$C_STA_MNT = 4     ; DDCMP is in maintenance mode
00000005 0000   532                 DDCMP$C_STA_FR1 = 5     ; There are three free states to
00000006 0000   533                 DDCMP$C_STA_FR2 = 6     ; keep the tables quadword
00000007 0000   534                 DDCMP$C_STA_FR3 = 7     ; aligned
0000    535
00000008 0000   536                 DDCMP$C_NUMSTA = 8      ; This value allows a quadword
0000    537                                                 ; entry per event will be difficult
0000    538                                                 ; to change
0000    539
00000005 0000   540                 DDCMP$C_ACTBITS = 5     ; Number of action bits per entry
00000003 0000   541                 DDCMP$C_STABITS = 3     ; Number of state bits per entry
000000E0 0000   542                 DDCMP$C_STAMSK  = <7>@5 ; State bit mask
0000    543
0000    544 ;
0000    545 ; Initialize the state transition table with default state and action
0000    546 ;
0000    547
0000    548 STTAB   HLT,BUG
0080    549
0080    550 EVENT   DDCMP$C_EV_RCV                  ; RCV a data message or an ACK, NAK
0000    551                                         ; or REP control mesage
```

```
0000    552
0000    553             STATE   HLT,    HLT,    IGNORE          ; Protocol in wrong state
0000    554             STATE   RUN,    RUN,    RCVMSG          ; Message was received
0000    555             STATE   IST,    IST,    SENDSTRT        ; Send the STRT request
0000    556             STATE   AST,    RUN,    RCVMSG          ; Start may be acknowledged
0000    557             STATE   MNT,    MNT,    IGNORE          ; Protocol in worng state
0000    558             STATE   FR1,    FR1,    BUG             ; Protocol should never be
0000    559             STATE   FR2,    FR2,    BUG             ; be in any of these states
0000    560             STATE   FR3,    FR3,    BUG
0000    561
0000    562 EVENT       DDCMP$C_EV_STI                  ; RCV a STRT message
0008    563
0008    564             STATE   HLT,    HLT,    IGNORE          ; User must start protocol
0008    565             STATE   RUN,    HLT,    HALT            ; Halt the protocol
0008    566             STATE   IST,    AST,    STARTACK        ; Wait for a start ack'd
0008    567             STATE   AST,    AST,    STARTACK        ; Wait for start ack'd
0008    568             STATE   MNT,    HLT,    HALT            ; Halt the protocol
0008    569             STATE   FR1,    FR1,    BUG             ; Protocol should never
0008    570             STATE   FR2,    FR2,    BUG             , be in any of these
0008    571             STATE   FR3,    FR3,    BUG             ; states
0008    572
0008    573 EVENT       DDCMP$C_EV_STA                  ; RCV a STACK message
0010    574
0010    575             STATE   HLT,    HLT,    MAINT           ; Protocol must be running
0010    576             STATE   RUN,    RUN,    RCVMSG          ; Receive the msg
0010    577             STATE   IST,    RUN,    RCVMSG          ; Receive the message
0010    578             STATE   AST,    RUN,    RCVMSG          ; Receive the message
0010    579             STATE   MNT,    MNT,    IGNORE          ; Ignore message
0010    580             STATE   FR1,    FR1,    BUG
0010    581             STATE   FR2,    FR2,    BUG
0010    582             STATE   FR3,    FR3,    BUG
0010    583
0010    584 EVENT       DDCMP$C_EV_RMM                  ; RCV maintenance message
0018    585
0018    586             STATE   HLT,    HLT,    IGNORE          ; Protocol is in wrong state
0018    587             STATE   RUN,    HLT,    HALT            ; Protocol is on wrong state
0018    588             STATE   IST,    MNT,    MAINT           ; Receive a maintenance message
0018    589             STATE   AST,    MNT,    MAINT           ; Receive a maintenance message
0018    590             STATE   MNT,    MNT,    MAINT           ; Receive a maintenance message
0018    591             STATE   FR1,    FR1,    BUG
0018    592             STATE   FR2,    FR2,    BUG
0018    593             STATE   FR3,    FR3,    BUG
0018    594
0018    595 EVENT       DDCMP$C_EV_XMT                  ; Transmit a message
0020    596
0020    597             STATE   HLT,    HLT,    IGNORE          ; Protocol is in wrong state
0020    598             STATE   RUN,    RUN,    XMTMSG          ; XMT a data or control msg
0020    599             STATE   IST,    IST,    STARTING        ; Protocol is in wrong state
0020    600             STATE   AST,    AST,    STARTING        ; Protocol is in wrong state
0020    601             STATE   MNT,    MNT,    MAINT           ; XMT a maint mg
0020    602             STATE   FR1,    FR1,    BUG
0020    603             STATE   FR2,    FR2,    BUG
0020    604             STATE   FR3,    FR3,    BUG
0020    605
0020    606 EVENT       DDCMP$C_EV_REP                  ; Message to reply timeout
0028    607
0028    608             STATE   HLT,    HLT,    IGNORE          ; Protocol is in wrong state
```

```
0028   609          STATE   RUN,    RUN,    REPTIM          ; Reply timeout the message
0028   610          STATE   IST,    IST,    IGNORE          ; Protocol is in wrong state
0028   611          STATE   AST,    AST,    IGNORE          ; Protocol is in wrong state
0028   612          STATE   MNT,    MNT,    CMPMNT          ; Complete the maint message
0028   613          STATE   FR1,    FR1,    BUG
0028   614          STATE   FR2,    FR2,    BUG
0028   615          STATE   FR3,    FR3,    BUG
0028   616
0028   617 EVENT    DDCMP$C_EV_SRT                  ; Start the protocol
0030   618
0030   619          STATE   HLT,    IST,    STARTINT        ; Enter to init the protocol
0030   620          STATE   RUN,    RUN,    IGNORE          ; Set wrong state for action
0030   621          STATE   IST,    IST,    IGNORE          ; Set wrong state for action
0030   622          STATE   AST,    AST,    IGNORE          ; Set wrong state for action
0030   623          STATE   MNT,    MNT,    IGNORE          ; Set wrong state for action
0030   624          STATE   FR1,    FR1,    BUG
0030   625          STATE   FR2,    FR2,    BUG
0030   626          STATE   FR3,    FR3,    BUG
0030   627
0030   628 EVENT    DDCMP$C_EV_STP                  ; Stop the protocol
0038   629
0038   630          STATE   HLT,    HLT,    HALT            ; Enter to stop the protocol
0038   631          STATE   RUN,    HLT,    HALT            ; Enter to stop the protocol
0038   632          STATE   IST,    HLT,    HALT            ; Enter to stop the protocol
0038   633          STATE   AST,    HLT,    HALT            ; Enter to stop the protocol
0038   634          STATE   MNT,    HLT,    HALT            ; Enter to stop the protocol
0038   635          STATE   FR1,    FR1,    BUG
0038   636          STATE   FR2,    FR2,    BUG
0038   637          STATE   FR3,    FR3,    BUG
0038   638
0038   639 EVENT    DDCMP$C_EV_QMT                  ; XMTQ is empty
0040   640
0040   641          STATE   HLT,    HLT,    IGNORE          ; Protocol is in wrong state
0040   642          STATE   RUN,    RUN,    QEMPTY          ; Does protocol have work
0040   643          STATE   IST,    IST,    IGNORE          ; Protocol is in wrong state
0040   644          STATE   AST,    AST,    QEMPTY          ; Does protocol have work
0040   645          STATE   MNT,    MNT,    IGNORE          ; Protocol is in wrong state
0040   646          STATE   FR1,    FR1,    BUG
0040   647          STATE   FR2,    FR2,    BUG
0040   648          STATE   FR3,    FR3,    BUG
0040   649
0040   650 EVENT    DDCMP$C_EV_MNT                  ; Enter maintenance mode
0048   651
0048   652          STATE   HLT,    MNT,    STARTMNT        ; Set protocol to maint mode
0048   653          STATE   RUN,    MNT,    STARTMNT        ; Set protocol to maint mode
0048   654          STATE   IST,    MNT,    STARTMNT        ; Set protocol to maint mode
0048   655          STATE   AST,    MNT,    STARTMNT        ; Set protocol to maint mode
0048   656          STATE   MNT,    MNT,    STARTMNT        ; Set protocol to maint mode
0048   657          STATE   FR1,    FR1,    BUG
0048   658          STATE   FR2,    FR2,    BUG
0048   659          STATE   FR3,    FR3,    BUG
0048   660
0048   661 EVENT    DDCMP$C_EV_ERR                  ; User requests error buffers
0050   662
0050   663          STATE   HLT,    HLT,    RETERR          ; Return error bfrs requested
0050   664          STATE   RUN,    RUN,    RETERR          ; Return error bfrs requested
0050   665          STATE   IST,    IST,    RETERR          ; Return error bfrs requested
```

```
0050   666          STATE   AST,    AST,    RETERR          ; Return error bfrs requested
0050   667          STATE   MNT,    MNT,    RETERR          ; Return error bfrs requested
0050   668          STATE   FR1,    FR1,    BUG
0050   669          STATE   FR2,    FR2,    BUG
0050   670          STATE   FR3,    FR3,    BUG
0050   671
0050   672 EVENT    DDCMP$C_EV_CHR                          ; Set line and station char
0058   673
0058   674          STATE   HLT,    HLT,    SETCHAR         ; Set char only when halted
0058   675          STATE   RUN,    RUN,    IGNORE          ; At any other time ignore
0058   676          STATE   IST,    IST,    IGNORE          ; At any other time ignore
0058   677          STATE   AST,    AST,    IGNORE          ; At any other time ignore
0058   678          STATE   MNT,    MNT,    IGNORE          ; At any other time ignore
0058   679          STATE   FR1,    FR1,    BUG
0058   680          STATE   FR2,    FR2,    BUG
0058   681          STATE   FR3,    FR3,    BUG
0058   682
0058   683 EVENT    DDCMP$C_EV_LNK                          ; Link was lost
0060   684
0060   685          STATE   HLT,    HLT,    IGNORE          ; Protocol was in wrong state
0060   686          STATE   RUN,    IST,    STARTINT        ; Will have to reinit
0060   687          STATE   IST,    IST,    STARTINT        ; Will have to reinit
0060   688          STATE   AST,    IST,    STARTINT        ; Will have to reinit
0060   689          STATE   MNT,    MNT,    IGNORE
0060   690          STATE   FR1,    FR1,    BUG
0060   691          STATE   FR2,    FR2,    BUG
0060   692          STATE   FR3,    FR3,    BUG
0060   693
0060   694 EVENT    DDCMP$C_EV_LER                          ; An error was detected on the
0068   695                                                  ; RVC'd message such that a NAK
0068   696                                                  ; must be sent
0068   697
0068   698          STATE   HLT,    HLT,    IGNORE
0068   699          STATE   RUN,    RUN,    SENDNAK         ; Send a NAK
0068   700          STATE   IST,    IST,    SENDNAK         ; Send a NAK
0068   701          STATE   AST,    AST,    SENDNAK         ; Send a NAK
0068   702          STATE   MNT,    MNT,    IGNORE
0068   703          STATE   FR1,    FR1,    BUG
0068   704          STATE   FR2,    FR2,    BUG
0068   705          STATE   FR3,    FR3,    BUG
0068   706
0068   707 EVENT    DDCMP$C_EV_STT                          ; Start DDCMP timer
0070   708
0070   709          STATE   HLT,    HLT,    START_TIMER     ; Start timer only when halted
0070   710          STATE   RUN,    RUN,    IGNORE          ; At any other time ignore
0070   711          STATE   IST,    IST,    IGNORE          ; At any other time ignore
0070   712          STATE   AST,    AST,    IGNORE          ; At any other time ignore
0070   713          STATE   MNT,    MNT,    IGNORE          ; At any other time ignore
0070   714          STATE   FR1,    FR1,    BUG
0070   715          STATE   FR2,    FR2,    BUG
0070   716          STATE   FR3,    FR3,    BUG
0070   717
0070   718 EVENT    DDCMP$C_EV_SPT                          ; Stop DDCMP timer
0078   719
0078   720          STATE   HLT,    HLT,    STOP_TIMER      ; Stop timer only when halted
0078   721          STATE   RUN,    RUN,    IGNORE          ; At any other time ignore
0078   722          STATE   IST,    IST,    IGNORE          ; At any other time ignore
```

```
                     0078   723          STATE   AST,    AST,    IGNORE          ; At any other time ignore
                     0078   724          STATE   MNT,    MNT,    IGNORE          ; At any other time ignore
                     0078   725          STATE   FR1,    FR1,    BUG
                     0078   726          STATE   FR2,    FR2,    BUG
                     0078   727          STATE   FR3,    FR3,    BUG
                     0078   728
                     0078   729 ENDSTTAB
                     0080   730
                     0080   731
                     0080   732 ; Local storage for CRC table
                     0080   733
00001401             0080   734 POLY:               .LONG   ^O12001
00000000             0084   735 CRCTABLE:           .LONG   ^X00000000
0000CC01             0088   736                     .LONG   ^X0000CC01
0000D801             008C   737                     .LONG   ^X0000D801
00001400             0090   738                     .LONG   ^X00001400
0000F001             0094   739                     .LONG   ^X0000F001
00003C00             0098   740                     .LONG   ^X00003C00
00002800             009C   741                     .LONG   ^X00002800
0000E401             00A0   742                     .LONG   ^X0000E401
0000A001             00A4   743                     .LONG   ^X0000A001
00006C00             00A8   744                     .LONG   ^X00006C00
00007800             00AC   745                     .LONG   ^X00007800
0000B401             00B0   746                     .LONG   ^X0000B401
00005000             00B4   747                     .LONG   ^X00005000
00009C01             00B8   748                     .LONG   ^X00009C01
00008801             00BC   749                     .LONG   ^X00008801
00004400             00C0   750                     .LONG   ^X00004400
                     00C4   751
                     00C4   752
```

```
                        00C4    754          .SBTTL  Command Interrupter
                        00C4    755
                        00C4    756 ;
                        00C4    757 ; The following routine acts as an interrupter between the commands that
                        00C4    758 ; the driver sends to DDCMP and the dispatch routine in this subroutine.
                        00C4    759 ; It takes the information passed to DDCMP via the register and decides
                        00C4    760 ; which event DDCMP is to perform.
                        00C4    761 ;
                        00C4    762 ;*****************************************************************************
                        00C4    763 ;       MUST LIST THE INTERFACE MEANINCS
                        00C4    764 ;       AND WHAT FMT ETC... STAND FOR
                        00C4    765 ;*****************************************************************************
                        00C4    766 ;
                        00C4    767 ;       INPUTS:
                        00C4    768 ;               R5 = Contain the address of the TF block
                        00C4    769 ;               R6 = Contains the command to DDCMP
                        00C4    770 ;               R7 = Contains the code pertaining to the command
                        00C4    771 ;               R8 = Address of the buffer to process
                        00C4    772 ;               R9 = Zero on input
                        00C4    773 ;               IPL = FIPL
                        00C4    774 ;
                        00C4    775 ;       OUTPUTS:
                        00C4    776 ;               R4 = Contain the event code
                        00C4    777 ;               R5 = Contain the address of the TF block
                        00C4    778 ;               R6, R7 & R9 must be preserved
                        00C4    779 ;               R8 = Address of the header if this is a receive msg
                        00C4    780 ;                    else it contains the addr of the buffer
                        00C4    781
                        00C4    782 DDCMP::
 53   4C A5    D0      00C4    783          MOVL    TF$A_GFB(R5),R3                 ; Get the global block address
                        00C8    784
      54        D4      00C8    785          CLRL    R4                              ; Clear the register
                        00CA    786
                        00CA    787          $DISPATCH       R6,TYPE=B,-
                        00CA    788          <-      ;index          ;action
                        00CA    789                  <DLK$C_RCVMSG   DMP$RCVMSG>-    ; Receive a message
                        00CA    790                  <DLK$C_XMTMSG   DMP$XMTMSG>-    ; Transmit a message
                        00CA    791                  <DLK$C_REQEBA   DMP$REQEBA>-    ; Request the error block
                        00CA    792                  <DLK$C_USRINT   DMP$USRINT>-    ; User initiates a state
                        00CA    793                  <DLK$C_CHAR     DMP$CHAR>-      ; Set trib char
                        00CA    794                  <DLK$C_START_TIMER DMP$STRTMER>- ; Start DDCMP timer
                        00CA    795                  <DLK$C_STOP_TIMER  DMP$STPTMER>- ; Stop DDCMP timer
                        00CA    796                  >
                        00DC    797
 57   04        3C      00DC    798          MOVZWL  #DLK$M_CMDERR,R7                ; If all else fails then
                        00DF    799                                                  ; set the error condition
      0183      31      00DF    800 10$:     BRW     DDCMP$REPORT                    ; Branch to report the bug
                        00E2    801
```

```
                        00E2    803
                        00E2    804 DMP$XMTMSG:
0A 57   01      E0      00E2    805             BBS     #DLK$V_QEMPTY,R7,10$            ; If BS then XMTQ empty detectd
0B 57   00      E0      00E6    806             BBS     #DLK$V_MSGSENT,R7,20$           ; If BS msg to reply T/O
   54   04      90      00EA    807             MOVB    #DDCMP$C_EV_XMT,R4              ; Set event to XMT a msg and
        00B3    31      00ED    808 5$:         BRW     DDCMP$EVENT                     ; jump to dispatch the event
   54   08      90      00F0    809 10$:        MOVB    #DDCMP$C_EV_QMT,R4              ; Q empty
        F8      11      00F3    810             BRB     5$
   54   05      90      00F5    811 20$:        MOVB    #DDCMP$C_EV_REP,R4              ; Message to reply timeout
        F3      11      00F8    812             BRB     5$
                        00FA    813
                        00FA    814 DMP$CHAR:
   54   0B      B0      00FA    815             MOVW    #DDCMP$C_EV_CHR,R4              ; Set event to set line char
        00A3    31      00FD    816             BRW     DDCMP$EVENT                     ; jump to dispatch the event
                        0100    817
                        0100    818 DMP$REQEBA:
   54   0A      B0      0100    819             MOVW    #DDCMP$C_EV_ERR,R4              ; Set event to retrieve err blk
        009D    31      0103    820             BRW     DDCMP$EVENT                     ; jump to dispatch the event
                        0106    821
                        0106    822 DMP$USRINT:
        57      95      0106    823             TSTB    R7                              ; If no char were set then
        1E      13      0108    824             BEQL    220$                            ; branch
                        010A    825
                        010A    826 ; Else test to determine which state the user wants
                        010A    827
06 57   02      E1      010A    828             BBC     #DLK$V_MAINT,R7,200$            ; If set then maint mode
   54   09      90      010E    829             MOVB    #DDCMP$C_EV_MNT,R4              ; set event and
        008F    31      0111    830             BRW     DDCMP$EVENT                     ; jump to dispatch the event
06 57   00      E1      0114    831 200$:       BBC     #DLK$V_START,R7,210$            ; If not BC then START protocl
   54   06      90      0118    832             MOVB    #DDCMP$C_EV_SRT,R4              ; set event and
        0085    31      011B    833             BRW     DDCMP$EVENT                     ; jump to dispatch the event
06 57   01      E1      011E    834 210$:       BBC     #DLK$V_STOP,R7,220$             ; If not BC then STOP protocol
   54   07      90      0122    835             MOVB    #DDCMP$C_EV_STP,R4              ; set event and
        007B    31      0125    836             BRW     DDCMP$EVENT                     ; jump to dispatch the event
                        0128    837 220$:
   57   01      3C      0128    838             MOVZWL  #DLK$M_BADSTATE,R7              ; set the error
        0137    31      012B    839             BRW     DDCMP$REPORT                    ; branch to report the error
                        012E    840
                        012E    841 DMP$STRTMER:
   54   0E      B0      012E    842             MOVW    #DDCMP$C_EV_STT,R4              ; Set event to start the timer
        006F    31      0131    843             BRW     DDCMP$EVENT                     ; jump to dispatch the event
                        0134    844
                        0134    845 DMP$STPTMER:
   54   0F      B0      0134    846             MOVW    #DDCMP$C_EV_SPT,R4              ; Set event to stop timer
        0069    31      0137    847             BRW     DDCMP$EVENT                     ; jump to dispatch the event
                        013A    848
```

```
                        013A   850
                        013A   851 DMP$RCVMSG:
          08    88      013A   852          BISB2   #TF$M_RCVDET,-                   ; Set that a message was
        16 A5           013C   853                  TF$B_SELTIM(R5)                  ; RCV'd
          57    95      013E   854          TSTB    R7                               ; If not eql error on RCV
          44    12      0140   855          BNEQ    130$
       05 68    91      0142   856          CMPB    MFD$B_MSGID(R8),#DDCMP$C_ENQ     ; Is the msg a control msg
          21    13      0145   857          BEQL    110$                             ; Branch if it is
    81 8F 68    91      0147   858          CMPB    MFD$B_MSGID(R8),#DDCMP$C_SOH     ; Is the msg a data msg
          34    13      014B   859          BEQL    120$                             ; Branch if it is
    90 8F 68    91      014D   860          CMPB    MFD$B_MSGID(R8),#DDCMP$C_DLE     ; Is the msg a maintenance msg
          0A    13      0151   861          BEQL    100$                             ; Branch if it is
          68    95      0153   862          TSTB    MFD$B_MSGID(R8)                  ; It is possible that no msg
                        0155   863                                                   ; was passed
          0B    13      0155   864          BEQL    108$                             ; Branch if that is the case
       57 08    3C      0157   865          MOVZWL  #DLK$M_RCVERR,R7                 ; Set the bit saying that an
                        015A   866                                                   ; unrecognized msg was received
          0108  31      015A   867          BRW     DDCMP$REPORT                     ; Branch to report the bug
                        015D   868
                        015D   869 ;
                        015D   870 ; Handle maintenance messages
                        015D   871 ;
       54 03    90      015D   872 100$:    MOVB    #DDCMP$C_EV_RMM,R4               ; Set maint msg received
          41    11      0160   873          BRB     DDCMP$EVENT                      ; jump to dispatch the event
                        0162   874
                        0162   875 ;
                        0162   876 ; Handle when no message was received
                        0162   877 ;
                        0162   878 108$:
       57 04    3C      0162   879          MOVZWL  #DLK$M_CMDERR,R7                 ; a null command hence
          00FD  31      0165   880          BRW     DDCMP$REPORT                     ; branch to report the error
                        0168   881
                        0168   882 ;
                        0168   883 ; Get the type of control field
                        0168   884 ;
                        0168   885
                        0168   886
 52   01 A8    9A      0168   887 110$:    MOVZBL  MFD$W_TYPFLG(R8),R2              ; Mov the msg subtype field
                        016C   888                                                   ; into R2
       06    52    91  016C   889          CMPB    R2,#DDCMP$C_STRT                 ; If the msg was not a STRT
          05    12      016F   890          BNEQ    115$                             ; Branch to next check else,
       54 01    90      0171   891          MOVB    #DDCMP$C_EV_STI,R4               ; set the reg. with the event
          2D    11      0174   892          BRB     DDCMP$EVENT                      ; & jump to dispatch the event
                        0176   893 ;
       07    52    91  0176   894 115$:    CMPB    R2,#DDCMP$C_STACK                ; If the msg was not a STACK
          06    12      0179   895          BNEQ    120$                             ; Branch to next check else,
       54 02    90      017B   896          MOVB    #DDCMP$C_EV_STA,R4               ; set the event and
          0022  31      017E   897          BRW     DDCMP$EVENT                      ; jump to dispatch the event
                        0181   898
       54 00    90      0181   899 120$:    MOVB    #DDCMP$C_EV_RCV,R4               ; For ACK, NAK, REP and all
          1D    11      0184   900          BRB     DDCMP$EVENT                      ; data msg set event and jump
                        0186   901
                        0186   902 ;
                        0186   903 ; If any errors were found then set which error and jump to record via
                        0186   904 ; the dispathcer.
                        0186   905 ;
                        0186   906
```

```
68    90 8F    91  0186   907 130$:   CMPB    #DDCMP$C_DLE,MFD$B_MSGID(R8)    ; If EQL DLE message
          0A   13  018A   908         BEQL    135$
0D 57     00   E1  018C   909         BBC     #DLK$V_LNKLOST,R7,140$          ; If BS then the link was lost
   54     0C   B0  0190   910         MOVW    #DDCMP$C_EV_LNK,R4              ; set event
        000D   31  0193   911         BRW     DDCMP$EVENT                     ; jump to dispatch the event
                   0196   912
51    2A A3    9E  0196   913 135$:   MOVAB   GF$B_MDF_CRC(R3),R1
      0F39     30  019A   914         BSBW    DDCMP$INCR_COUNTER
   54    0D    90  019D   915 140$:   MOVB    #DDCMP$C_EV_LER,R4              ; Assume for all other cases
                   01A0   916                                                 ; an NAK needs to be sent
                   01A0   917                                                 ; althougth not always
        0000     31  01A0   918         BRW     DDCMP$EVENT
                   01A3   919
```

```
                        01A3   921          .SBTTL  Common event dispatcher
                        01A3   922
                        01A3   923 ;
                        01A3   924 ; DDCMP$EVENT - Common event dispatcher
                        01A3   925 ;
                        01A3   926 ; FUNCTIONAL DESCRIPTION
                        01A3   927 ;
                        01A3   928 ; This is the common state table event dispatcher used to determine what
                        01A3   929 ; is to be done and what state DDCMP is to enter next. An event only has
                        01A3   930 ; meaning within the context of DDCMP.
                        01A3   931 ;
                        01A3   932 ;       INPUTS:
                        01A3   933 ;               R9 = Data area for interface with driver
                        01A3   934 ;               R8 = Contains the address of the rcv msg or it
                        01A3   935 ;                    contains the address of the transmit buffer
                        01A3   936 ;               R7 = Contains codes pertinent to the routine to call.
                        01A3   937 ;               R6 = Commands from the driver
                        01A3   938 ;               R5 = Must contain the address of the TF block
                        01A3   939 ;               R4 = Event code to be processed
                        01A3   940 ;               R3 = Global field block address
                        01A3   941 ;               R2 = Available for dispatcher's exclusive use
                        01A3   942 ;               R1 = scratch
                        01A3   943 ;               R0 = scratch
                        01A3   944 ;
                        01A3   945 ;
                        01A3   946 ;       OUTPUTS:
                        01A3   947 ;               R6,R7 and R9 = Contain information for interface
                        01A3   948 ;               R5,R8 are preserved
                        01A3   949 ;               R0 = Contains status code from action routine
                        01A3   950 ;               All other registers are destroyed
                        01A3   951 ;
                        01A3   952 ;
                        01A3   953
                        01A3   954          ASSUME DDCMP$C_NUMSTA EQ 8              ; Assume quadword per event
                        01A3   955
                        01A3   956 DDCMP$EVENT:
          52    63  9A  01A3   957          MOVZBL  GF$B_STATE(R3),R2               ; Get state of protocol
      48 A5     58  D0  01A6   958          MOVL    R8,TF$A_BUFPTR(R5)              ; Save the addr of the buffer
   51  FE51 CF44  7E  01AA   959          MOVAQ   DDCMP$AB_STTAB[R4],R1           ; Get event block address
       52   6142  90  01B0   960          MOVB    (R1)[R2],R2                     ; Get table entry
   51  52   E0 8F  8B  01B4   961          BICB3   #DDCMP$C_STAMSK,R2,R1           ; Get action routine index
            52      DD  01B9   962          PUSHL   R2
                        01BB   963 ;
                        01BB   964 ;
                        01BB   965 ; Dispatch to the event code. The routine can assume the following
                        01BB   966 ;
                        01BB   967 ;       R9 = Data from the interface
                        01BB   968 ;       R8 = Address of received message header
                        01BB   969 ;       R7 = Data from the interface
                        01BB   970 ;       R5 = TF block address
                        01BB   971 ;       R4 = Event code
                        01BB   972 ;       R3 = Global field block address
                        01BB   973 ;       (SP) = Return address
                        01BB   974 ;       4(SP) = R2 which contains the state inforamtion for the dispatch
                        01BB   975 ;
                        01BB   976 ; Returned values
                        01BB   977 ;       R3 = Address of the GF block
```

```
                     01BB   978 ;       R5 = Address of the TF block
                     01BB   979 ;       R9 - R6 = Information to pass to driver on RSB
                     01BB   980 ;
                     01BB   981 ;
                     01BB   982
000001F3'EF    DF    01BB   983          PUSHAL  CHANGE_STA                      ; Setup return address
                     01C1   984          $DISPATCH       R1,TYPE=B,-
                     01C1   985          <-      ;index              action
                     01C1   986
                     01C1   987                  <ACT$_IGNORE        DDCMP$IGNORE>-       ; Ignore event
                     01C1   988                  <ACT$_SENDSTRT      DDCMP$SENDSTRT>-     ; Send STRT message
                     01C1   989                  <ACT$_BUG           DDCMP$BUG>-          ; State transtion bug
                     01C1   990                  <ACT$_SETCHAR       DDCMP$SETCHAR>-      ; Set lne & staton char
                     01C1   991                  <ACT$_STARTINT      DDCMP$STARTINT>-     ; Init a start on prot
                     01C1   992                  <ACT$_STARTACK      DDCMP$STARTACK>-     ; Ack a start on prot
                     01C1   993                  <ACT$_STARTMNT      DDCMP$STARTMNT>-     ; Enter maint mode
                     01C1   994                  <ACT$_RCVMSG        DDCMP$RCVMSG>-       ; Rcv a message
                     01C1   995                  <ACT$_XMTMSG        DDCMP$XMTMSG>-       ; Xmt a message
                     01C1   996                  <ACT$_REPTIM        DDCMP$REPTIM>-       ; Reply T/O the message
                     01C1   997                  <ACT$_QEMPTY        DDCMP$QEMPTY>-       ; XMTQ is empty
                     01C1   998                  <ACT$_SENDNAK       DDCMP$SENDNAK>-      ; Send a NAK
                     01C1   999                  <ACT$_HALT          DDCMP$HALT>-         ; Stop the protocol
                     01C1  1000                  <ACT$_MAINT         DDCMP$MAINT>-        ; Run in maint mode
                     01C1  1001                  <ACT$_CMPMNT        DDCMP$CMPMNT>-       ; Comp maint message
                     01C1  1002                  <ACT$_RETERR        DDCMP$RETERR>-       ; Return error buffer
                     01C1  1003                  <ACT$_STARTING      DDCMP$STARTING>-     ; Protocol is starting
                     01C1  1004                  <ACT$_START_TIMER   DDCMP$START_TIMER>-  ; Start DDCMP timer
                     01C1  1005                  <ACT$_STOP_TIMER    DDCMP$STOP_TIMER>-   ; Stop DDCMP timer
                     01C1  1006                  >
          56   09   3C   01EB  1007          MOVZWL  S^#DLK$C_ACTNOTCOM,R6          ; Set up return status for err
          57   02   3C   01EE  1008          MOVZWL  #DLK$M_BADACTION,R7
               3B   11   01F1  1009          BRB     DDCMP$ERROR
                         01F3  1010
                         01F3  1011 CHANGE_STA:
               52 8ED0   01F3  1012          POPL    R2
          58   48 A5     D0   01F6  1013          MOVL    TF$A_BUFPTR(R5),R8          ; Restore the buffer address
                    05   EF   01FA  1014          EXTZV   #DDCMP$C_ACTBITS,-
        51    52   03         01FC  1015                  #DDCMP$C_STABITS,R2,R1      ; Get next state
                         01FF  1016
                         01FF  1017 ; Because we can not always assure that tranmsits will to a byte
                         01FF  1018 ; count run out in ASYNCH DDCMP mode, we will check here that
                         01FF  1019 ; a maintenance messages has bee validated before chaning the driver
                         01FF  1020 ; state to maintenance mode.
                         01FF  1021
               51   04   91   01FF  1022          CMPB    #DDCMP$C_STA_MNT,R1         ; Change to maint mode
                    0D   12   0202  1023          BNEQ    5$                          ; BNEQ if no
          09 57     10   E1   0204  1024          BBC     #DLK$V_CRC,R7,5$            ; If BC then message ok make change
57   00000080 8F     CA   0208  1025          BICL    #DLK$M_MNTRCV,R7            ; Else do not change to mop mode
                    0D   11   020F  1026          BRB     10$
               63   51   91   0211  1027 5$:     CMPB    R1,GF$B_STATE(R3)           ; New state?
                    08   13   0214  1028          BEQL    10$
               63   51   90   0216  1029          MOVB    R1,GF$B_STATE(R3)           ; Change state
                         0219  1030
                         0219  1031          ASSUME  DDCMP$C_STA_RUN EQ 1
                         0219  1032          ASSUME  DDCMP$C_STA_IST EQ 2
                         0219  1033          ASSUME  DDCMP$C_STA_AST EQ 3
                         0219  1034
```

```
03   51   91  0219  1035          CMPB    R1,S^#DDCMP$C_STA_AST           ; If not one a change to RUN,
     01   15  021C  1036          BLEQ    20$                             ; IST or AST then fall thru
                021E  1037
               05  021E  1038 10$:    RSB                                 ; Return to driver
                021F  1039
                021F  1040 ; Clear error counters
00   00   F0  021F  1041 20$:    INSV    #0,#DDCMP$C_XMT_THRS,-          ; Clear XMT thrs cnter
     03             0222  1042                  #DDCMP$C_THR_SIZE,-
01C5 C5             0223  1043                  TF$W_THRES(R5)
03   00   F0  0226  1044          INSV    #0,#DDCMP$C_RCV_THRS,-          ; Clear RCV thrs cnter
     03             0229  1045                  #DDCMP$C_THR_SIZE,-
01C5 C5             022A  1046                  TF$W_THRES(R5)
               05  022D  1047          RSB
                022E  1048
                022E  1049
```

```
                     022E  1051
                     022E  1052          .SBTTL  Error
                     022E  1053
                     022E  1054 ;DDCMP$ERROR
                     022E  1055 ; This routine clears DDCMP$EVENT return address from the stack
                     022E  1056 ; so that control will immediately return to the driver. This is
                     022E  1057 ; done when header or data errors are found in a message. It
                     022E  1058 ; is also called when the protocol finds that it has received
                     022E  1059 ; a ENQ other then what it expected so as not to change the state
                     022E  1060 ; of the protocol.
                     022E  1061 ;
                     022E  1062 ;       INPUTS          R5-R9 set up with error status to driver.
                     022E  1063 ;                       (SP)  = Address of CHANGE_STA
                     022E  1064 ;                       4(SP) = R2
                     022E  1065 ;                       8(SP) = The return address to the driver
                     022E  1066 ;
                     022E  1067 ;
                     022E  1068 DDCMP$ERROR:
   C2 AF    9F       022E  1069          PUSHAB  B^CHANGE_STA                    ; Save the address expected
8E    8E    D1       0231  1070          CMPL    (SP)+,(SP)+                     ; If NEQ then bugcheck
      04    12       0234  1071          BNEQ    10$
      52 8ED0        0236  1072          POPL    R2                              ; Clear the register from stack
         05          0239  1073          RSB                                     ; Return to driver
                     023A  1074
                     023A  1075 10$:     BUG_CHECK       NOBUFPCKT,FATAL
                     023E  1076
```

```
                     023E  1078
                     023E  1079          .SBTTL  Command error handling routines
                     023E  1080 ; IGNORE
                     023E  1081 ;
                     023E  1082 ; This routine is called when a request is made to the protocol, but the
                     023E  1083 ; protocol is in the wrong state to complete the action.
                     023E  1084 ;
                     023E  1085 ;       INPUTS          none
                     023E  1086 ;
                     023E  1087 ;       OUTPUTS         R6 = Contains the status code to return to the driver
                     023E  1088 ;                       R7 = Contains the reason for the abort
                     023E  1089 ;
                     023E  1090 DDCMP$IGNORE:
56   09   3C         023E  1091          MOVZWL  S^#DLK$C_ACTNOTCOM,R6           ; Set the abort status
57   01   3C         0241  1092          MOVZWL  #DLK$M_BADSTATE,R7              ; Set the reason for the abort
               05    0244  1093          RSB
                     0245  1094
                     0245  1095
                     0245  1096 ; STARTING
                     0245  1097 ;
                     0245  1098 ; This routine is called when a request to transmit a message is made and
                     0245  1099 ; the protocol has not finished its start up sequence. It fakes a queue
                     0245  1100 ; full error and puts the transmit on the overflow queue.
                     0245  1101 ;
                     0245  1102 ;       INPUTS          P3 = Address of the GF block
                     0245  1103 ;                       R5 = Address of the TF block
                     0245  1104 ;                       R8 = Address of data message
                     0245  1105 ;
                     0245  1106 ;       OUTPUTS         R3,R5 are preserved
                     0245  1107 ;                       R6 = Contains the status code to driver
                     0245  1108 ;                       R7 = Contains the reason code
                     0245  1109 ;
                     0245  1110 DDCMP$STARTING:
   56   08   3C       0245  1111          MOVZWL  S^#DLK$C_RETINFO,R6             ; Return status
   57   20   3C       0248  1112          MOVZWL  #DLK$M_QFULERR,R7               ; Fake a queue full error
             58   D5  024B  1113          TSTL    R8                              ; If eql then no message
             0A   13  024D  1114          BEQL    10$
13    0A     A8   91  024F  1115          CMPB    XMTQ$B_BUFTYP(R8),S^#DYN$C_BUFIO ; If NEQ then not correct buf
             05   12  0253  1116          BNEQ    20$
44 B5        68   0E  0255  1117          INSQUE  (R8),@TF$Q_XMTOVF+4(R5)         ; Insert at the end of queue
                     0259  1118 10$:      RSB
                     025A  1119
                     025A  1120 20$:      BUG_CHECK       NOBUFPCKT,FATAL
                     025E  1121
```

```
                     025E  1123
                     025E  1124 ; BUG and REPORT
                     025E  1125 ;
                     025E  1126 ; This routine is caled when the state driven portion of the protocol is
                     025E  1127 ; requested to do an action that is doesn't know or isn't allowed to do.
                     025E  1128 ;
                     025E  1129 ;       INPUTS          REPORT - R7 contains the reason for the abort
                     025E  1130 ;
                     025E  1131 ;       OUTPUTS         R6 = Return code to driver - Action not complete
                     025E  1132 ;                       R7 = Reason that the action wasn't completed
                     025E  1133 ;
                     025E  1134 DDCMP$BUG:
57   02   3C         025E  1135         MOVZWL  #DLK$M_BADACTION,R7             ; Set up reason for incomp act
                     0261  1136         BUG_CHECK NOBUFPCKT,FATAL
                     0265  1137
                     0265  1138 DDCMP$REPORT:
56   09   3C         0265  1139         MOVZWL  S^#DLK$C_ACTNOTCOM,R6           ; Set up reg with status for
                     0268  1140                                                 ; return to driver
                05   0268  1141         RSB
                     0269  1142
```

```
0269  1144
0269  1145          .SBTTL  Set charateristics
0269  1146
0269  1147 ; DDCMP$SETCHAR - Set Charateristics
0269  1148 ;
0269  1149 ; This routine sets up charateristics for the protocol.  This should be
0269  1150 ; called as least once by the driver to set up default characteristics on
0269  1151 ; the station.
0269  1152 ;
0269  1153 ; The routine works as follows. It is called by the driver only when the
0269  1154 ; protocol is in halt state. Which means at Unit init time or after the
0269  1155 ; protocol has been put in HALT state for whatever reason. If the set
0269  1156 ; default bit is set then the characteristics byte is cleared and the
0269  1157 ; protocol is set up for DDCMP point to point full duplex mode. If it is
0269  1158 ; not clear then the protocol assumes that the user wishes to set some
0269  1159 ; parameters and those parameters are set. This routine must be called
0269  1160 ; at least once to set defaults for all settable DDCMP parameters. Defaults
0269  1161 ; for the paramters are determined by the driver using this protocol.
0269  1162 ;
0269  1163 ;       Example defaults:
0269  1164 ;
0269  1165 ;                               TF$B_ADDR  = 1
0269  1166 ;                               TF$B_XADDR = 1
0269  1167 ;                               TF$B_RADDR = 1
0269  1168 ;                               GF$W_SELWAI = 3 seconds
0269  1169 ;                               GF$B_MAXSEL = 2 tries
0269  1170 ;                               TF$W_REPWAI = 3 seconds
0269  1171 ;                               TF$B_MAXRTO = 4 tries
0269  1172 ;                               TF$B_MSGCNT = 4 per selection interval
0269  1173 ;                               GF$B_STRTIM = 1 second
0269  1174 ;                               GF$B_BABTIM = 1 second
0269  1175 ;
0269  1176 ;       INPUTS                  R3 = Address of the GF block
0269  1177 ;                               R5 = Address of TF block
0269  1178 ;                               R6 = Command from driver
0269  1179 ;                               R7 = Fields to set
0269  1180 ;                               R8 = Address of block containing parameters
0269  1181 ;                               R9 = Address of a block containing the addresses the
0269  1182 ;                                    protocol needs from the driver to do the
0269  1183 ;                                    retransmission of messages properly.
0269  1184 ;
0269  1185 ;       OUPUTS                  R3,R5 are preserved
0269  1186 ;                               R6 = Contain return status to driver.
0269  1187 ;                               R7 = Normal return
0269  1186 ;                               R8,R9 are destroyed
0269  1189 ;
0269  1190 DDCMP$SETCHAR:
        03 57   0B   E4  0269  1191          BBSC    #DLK$V_SETDEF,R7,2$              ; If BS defaults to set
              0090   31  026D  1192          BRW     5$                               ; Else branch to set parameters
           0C A5     94  0270  1193 2$:      CLRB    TF$B_CHAR(R5)                    ; Set defaults of zero
                         0273  1194
                         0273  1195          ASSUME  TF$B_XADDR EQ TF$B_ADDR+1
                         0273  1196          ASSUME  TF$B_RADDR EQ TF$B_XADDR+1
                         0273  1197
OD A5   00010101   8F   D0  0273  1198          MOVL    #65793,TF$B_ADDR(R5)             ; Set all default addr to 1
                01   90  027B  1199          MOVB    #DDCMP$C_REALCLOCK,-             ; Set the default for reply T/O
             17 A5         027D  1200                  TF$B_REPTIM(R5)                  ; to be the real time clock
```

```
                     027F  1201
                1B A5  94  027F  1202          CLRB    TF$B_CURRTO(R5)                 ; Clear reply timeout and sel't
                08 A3  94  0282  1203          CLRB    GF$B_CURSEL(R3)                 ; fields only once per protocol
                           0285  1204                                                  ; startup
                           0285  1205
                           0285  1206 ; If R9 is set up then it will contain the address that the protocol needs
                           0285  1207 ; to perform message retransmission properly.
                           0285  1208
                   59  D5  0285  1209          TSTL    R9
                   09  13  0287  1210          BEQL    3$
             54 A5 69  D0  0289  1211          MOVL    DLA$A_XMT_INPR(R9),TF$A_XMT_INPR(R5)
          58 A5  04 A9  D0  028D  1212          MOVL    DLA$A_POSTQ(R9),TF$A_POSTQ(R5)
                           0292  1213
                           0292  1214 ; Set up the NMA def's for counters
                           0292  1215 ;
                           0292  1216 ; Global counters
                           0292  1217
                           0292  1218 3$:       COUNT   NMA$C_CTLIN_LPE,-
                           0292  1219                  BITMAP=YES,-
                           0292  1220                  WIDTH=8,-
                           0292  1221                  LOC=GF$W_LSETYP(R3)
                           0298  1222
                           0298  1223          COUNT   NMA$C_CTLIN_RPE,-
                           0298  1224                  BITMAP=YES,-
                           0298  1225                  WIDTH=8,-
                           0298  1226                  LOC=GF$W_RSETYP(R3)
                           029E  1227
                           029E  1228 ;
                           029E  1229 ; Tributary counters
                           029E  1230 ;
                           029E  1231
                           029E  1232          COUNT   NMA$C_CTCIR_DEO,-
                           029E  1233                  BITMAP=YES,-
                           029E  1234                  WIDTH=8,-
                           029E  1235                  LOC=TF$W_DEOTYP(R5)
                           02A5  1236
                           02A5  1237          COUNT   NMA$C_CTCIR_DEI,-
                           02A5  1238                  BITMAP=YES,-
                           02A5  1239                  WIDTH=8,-
                           02A5  1240                  LOC=TF$W_DEITYP(R5)
                           02AC  1241
                           02AC  1242          COUNT   NMA$C_CTCIR_LRT,-
                           02AC  1243                  WIDTH=8,-
                           02AC  1244                  LOC=TF$W_LRTOTYP(R5)
                           02B3  1245
                           02B3  1246          COUNT   NMA$C_CTCIR_RRT,-
                           02B3  1247                  WIDTH=8,-
                           02B3  1248                  LOC=TF$W_RRTOTYP(R5)
                           02BA  1249
                           02BA  1250          COUNT   NMA$C_CTCIR_SLT,-
                           02BA  1251                  BITMAP=YES,-
                           02BA  1252                  WIDTH=8,-
                           02BA  1253                  LOC=TF$W_STOTYP(R5)
                           02C1  1254
                           02C1  1255          COUNT   NMA$C_CTCIR_LBE,-
                           02C1  1256                  BITMAP=YES,-
                           02C1  1257                  WIDTH=8,-
```

```
                02C1  1258                      LOC=TF$W_LBETYP(R5)
                02C8  1259
                02C8  1260          COUNT       NMA$C_CTCIR_RBE,-
                02C8  1261                      BITMAP=YES,-
                02C8  1262                      WIDTH=8,-
                02C8  1263                      LOC=TF$W_RBETYP(R5)
                02CF  1264
                02CF  1265          COUNT       NMA$C_CTCIR_SIE,-
                02CF  1266                      WIDTH=16,-
                02CF  1267                      LOC=TF$W_SELTYP(R5)
                02D6  1268
                02D6  1269          COUNT       NMA$C_CTCIR_DBS,-
                02D6  1270                      WIDTH=32,-
                02D6  1271                      LOC=TF$W_DMXTYP(R5)
                02DD  1272
                02DD  1273          COUNT       NMA$C_CTCIR_DBR,-
                02DD  1274                      WIDTH=32,-
                02DD  1275                      LOC=TF$W_DMRTYP(R5)
                02E4  1276
                02E4  1277          COUNT       NMA$C_CTCIR_BSN,-
                02E4  1278                      WIDTH=32,-
                02E4  1279                      LOC=TF$W_DBXTYP(R5)
                02EB  1280
                02EB  1281          COUNT       NMA$C_CTCIR_BRC,-
                02EB  1282                      WIDTH=32,-
                02EB  1283                      LOC=TF$W_DBRTYP(R5)
                02F2  1284
                02F2  1285 ; Set up the size for the counter blocks
                02F2  1286
1E A5   01C5 8F   018A 8F   A3  02F2  1287          SUBW3   #TF$K_ERRSTRT,#TF$K_ERREND,TF$W_TEB(R5)
     0B A3   29   1F   A3   02FB  1288          SUBW3   #GF$K_ERRSRT,#GF$K_ERREND,GF$W_GEB(R3)
                                0300  1289
                                0300  1290          ASSUME  DLK$V_STATYP EQ 0
                                0300  1291          ASSUME  DLK$V_POINT EQ 1
                                0300  1292          ASSUME  DLK$V_DUPLEX EQ 2
                                0300  1293          ASSUME  DLK$V_LNTYP EQ 3
                                0300  1294          ASSUME  DLK$V_MSGCNT EQ 4
                                0300  1295          ASSUME  DLK$V_SELTIM EQ 5
                                0300  1296          ASSUME  DLK$V_SELWAIT EQ 6
                                0300  1297          ASSUME  DLK$V_REPTIM EQ 7
                                0300  1298          ASSUME  DLK$V_REPWAIT EQ 8
                                0300  1299          ASSUME  DLK$V_STREAM EQ 9
                                0300  1300          ASSUME  DLK$V_BABBLE EQ 10
                                0300  1301
                     50   D4   0300  1302 5$:      CLRL    R0                      ; Clear the counter field
                  08 57   E9   0302  1303          BLBC    R7,10$                  ; If BC then first bit not set
          00000030D'EF   DF   0305  1304          PUSHAL  10$                     ; Push this address to loop
                                030B  1305                                          ; thru the case to test all
                                030B  1306                                          ; bits that may be set
                     12   11   030B  1307          BRB     15$                     ; Branch to CASE
                0A   50   91   030D  1308 10$:     CMPB    R0,#DLK$V_BABBLE        ; Have all setable bits been
                     0D   14   0310  1309          BGTR    15$                     ; set. If GTR then yes they
                                0312  1310                                          ; Have branch to fall thru CASE
   57   57   FF 8F   78   0312  1311          ASHL    #-1,R7,R7               ; Get next bit
                     50   96   0317  1312          INCB    R0
                  F1 57   E9   0319  1313          BLBC    R7,10$                  ; If BC bit not set
                  EE AF   DF   031C  1314          PUSHAL  10$                     ; Push return address
```

```
                     031F  1315                                                   ; Fall thru to the CASE
                     031F  1316
                     031F  1317 ; Please note that the way this routine sets up the protocol characteristics
                     031F  1318 ; this CASE statement must be a CASE statement and can NOT be changes
                     031F  1319 ; to a $DISPATCH.
                     031F  1320
                     031F  1321 15$:     CASE    R0,<-                          ; Test to see which bit is set
                     031F  1322                  SET$STATYP,-                   ; jump to the routine to handle
                     031F  1323                  SET$POINT,-                    ; that value
                     031F  1324                  SET$DUPLEX,-
                     031F  1325                  SET$LNTYP,-
                     031F  1326                  SET$MSGCNT,-
                     031F  1327                  SET$SELTIM,-
                     031F  1328                  SET$SELWAIT,-
                     031F  1329                  SET$REPTIM,-
                     031F  1330                  SET$REPWAIT,-
                     031F  1331                  >,LIMIT=#DLK$V_STATYP
                     0335  1332
      0C A5    95    0335  1333          TSTB    TF$B_CHAR(R5)                  ; If NEQ then not FPC, branch
         04    12    0338  1334          BNEQ    20$
         01    90    033A  1335          MOVB    S^#DDCMP$C_REALCLOCK,-         ; Move in type of reply timeout
      17 A5          033C  1336                  TF$B_REPTIM(R5)                ; clock - real time for FPC
   56    08    3C    033E  1337 20$:     MOVZWL  S^#DLK$C_RETINFO,R6            ; Set normal return status
         58    7C    0341  1338          CLRQ    R8
               05    0343  1339          RSB
                     0344  1340
                     0344  1341 ; Start of routines to branch to via the CASE on R0
                     0344  1342
                     0344  1343
                     0344  1344 SET$STATYP:
   0C A5     03    88  0344  1345          BISB2   #<-                            ; Set that this is a Tributary
                     0348  1346                  TF$M_STATYP!-                  ; station and that it is multi-
                     0348  1347                  TF$M_POINT-                    ; point
                     0348  1348                  >,TF$B_CHAR(R5)
   0D A5     68    90  0348  1349          MOVB    DLK$B_TRIB(R8),TF$B_ADDR(R5)   ; Set the trib address
   0E A5     68    90  034C  1350          MOVB    DLK$B_TRIB(R8),TF$B_XADDR(R5)  ; Set the trib address for XMTS
   0F A5     68    90  0350  1351          MOVB    DLK$B_TRIB(R8),TF$B_RADDR(R5)  ; Set the trib address for RCVS
         03    90  0354  1352          MOVB    S^#DDCMP$C_NXTINTVL,-          ; Move in type of reply timer
      17 A5          0356  1353                  TF$B_REPTIM(R5)                ; Next selection interval
               05    0358  1354          RSB
                     0359  1355
                     0359  1356 ;
                     0359  1357 ; Assume if gotten here then the station is a control station
                     0359  1358 ;
                     0359  1359 SET$POINT:
         02    88  0359  1360          BISB2   #TF$M_POINT,-                  ; Set that the station is
      0C A5          035B  1361                  TF$B_CHAR(R5)                  ; multipoint
         02    90  035D  1362          MOVB    S^#DDCMP$C_EOINTVL,-           ; Set the reply timer to be the
      17 A5          035F  1363                  TF$B_REPTIM(R5)                ; end of the next selection
                     0361  1364                                                 ; interval
               05    0361  1365          RSB
                     0362  1366
                     0362  1367 SET$DUPLEX:
         04    88  0362  1368          BISB2   #TF$M_DUPLEX,-                 ; Set that the line is half
      0C A5          0364  1369                  TF$B_CHAR(R5)                  ; duplex
         00    E0  0366  1370          BBS     #TF$V_STATYP,-                 ; Branch if BS station is a
   04 0C A5          0368  1371                  TF$B_CHAR(R5),5$               ; trib station
```

```
                     03   90   036B  1372          MOVB    S^#DDCMP$C_NXTINTVL,-           ; Set reply timer to be the
                 17  A5                1373                  TF$B_REPTIM(R5)                 ; next selection interval
                          05   036F  1374 5$:      RSB
                               0370  1375
                               0370  1376 SET$LNTYP:
              02 A3    01    88   0370  1377          BISB2   #GF$M_CRC,GF$B_DRVCHR(R3)       ; Set device does not do CRC checks
                          05   0374  1378          RSB
                               0375  1379
                               0375  1380 SET$MSGCNT:
                 01  A8    90   0375  1381          MOVB    DLK$B_MSGCNT(R8),-              ; Set no msg per selection
                 1C  A5               0378  1382                TF$B_MSGCNT(R5)
                          05   037A  1383 5$:      RSB
                               037B  1384
                               037B  1385 SET$SELTIM:
     07 A3    01    03  A8    81   037B  1386          ADDB3   DLK$B_MAXSEL(R8),#1,GF$B_MAXSEL(R3) ; Move in the # of times
     2B A3    05    07  A3    85   0381  1387          MULB3   GF$B_MAXSEL(R3),#5,GF$B_STRTSEL(R3) ; Move in the # of sels for star
                               0387  1388                                                  ; the select will T/O field
                          05   0387  1389          RSB
                               0388  1390
                               0388  1391 SET$SELWAIT:
                     52   DD   0388  1392          PUSHL   R2
              16 0C  A5    00   E0   038A  1393          BBS     #TF$V_STATYP,TF$B_CHAR(R5),5$   ; If trib set number on
                               038F  1394                                                  ; interv before deselection
               52    03E8  8F   3C   038F  1395          MOVZWL  #DDCMP$C_MILLI,R2               ; Get the size to div by
                     52   B7   0394  1396          DECW    R2                              ; Sub 1 from it
                  52   06  A8   A0   0396  1397          ADDW    DLK$W_SELWAIT(R8),R2            ; Ensure time is at least 1 sec
     03 A3    52    03E8  8F   A7   039A  1398          DIVW3   #DDCMP$C_MILLI,R2,GF$W_SELWAI(R3) ; Set the timer in seconds
                     52 8ED0   03A1  1399          POPL    R2
                          05   03A4  1400          RSB
                               03A5  1401
               52    00FA  8F   3C   03A5  1402 5$:      MOVZWL  #DDCMP$C_QUA,R2                 ; Get the size to div by
                     52   B7   03AA  1403          DECW    R2                              ; Sub 1 from it
                  52   06  A8   A0   03AC  1404          ADDW    DLK$W_SELWAIT(R8),R2            ; Ensure time is at least 1 sec
     03 A3    52    00FA  8F   A7   03B0  1405          DIVW3   #DDCMP$C_QUA,R2,GF$W_SELWAI(R3) ; Set the timer in seconds
                     52 8ED0   03B7  1406          POPL    R2
                          05   03BA  1407          RSB
                               03BB  1408
                               03BB  1409 SET$REPTIM:
                 02  A8    81   03BB  1410          ADDB3   DLK$B_MAXREP(R8),-              ; Set max reply T/O
                  1A A5    01              03BE  1411                #1,TF$B_MAXRTO(R5)
                          05   03C1  1412          RSB
                               03C2  1413
                               03C2  1414 SET$REPWAIT:
                     52   DD   03C2  1415          PUSHL   R2
               52    03E8  8F   3C   03C4  1416          MOVZWL  #DDCMP$C_MILLI,R2               ; Get the size to div by
                     52   B7   03C9  1417          DECW    R2                              ; Sub 1 from it
                  52   04  A8   A0   03CB  1418          ADDW    DLK$W_REPWAIT(R8),R2            ; Ensure time is at least 1 sec
     18 A5    52    03E8  8F   A7   03CF  1419          DIVW3   #DDCMP$C_MILLI,R2,TF$W_REPWAI(R5); Set the timer in seconds
                     52 8ED0   03D6  1420          POPL    R2
                          05   03D9  1421          RSB
                               03DA  1422
```

```
                        03DA  1424
                        03DA  1425          .SBTTL  Initate start of protocol
                        03DA  1426
                        03DA  1427 ; STARTINT - Initate start
                        03DA  1428 ;
                        03DA  1429 ; This routine initiates a start of the protocol. It expects that the char for
                        03DA  1430 ; the protocol have been set or default elsewhere. If an error is found in
                        03DA  1431 ; the header passed to the protocol then return immediately to the driver
                        03DA  1432 ; to report a header error.
                        03DA  1433 ;
                        03DA  1434 ;         INPUT           R3 = Address of the GF block
                        03DA  1435 ;                         R5 = Address of the TF block
                        03DA  1436 ;                         R8 = Address of the message header if the start was
                        03DA  1437 ;                              initated by a remote node
                        03DA  1438 ;
                        03DA  1439 ;         OUTPUTS         R3, R5 and R8 are preserved
                        03DA  1440 ;                         R6,R7 contain the return status for driver
                        03DA  1441 ;                         XMTQ is given an entry
                        03DA  1442 ;                         RTOQ is given an entry
                        03DA  1443          .ENABL  LSB
                        03DA  1444 DDCMP$SENDSTRT:
56   08   3C   03DA  1445          MOVZWL  S^#DLK$C_RETINFO,R6             ; Set up return to driver
     57   D4   03DD  1446          CLRL    R7                              ; Clear all set bits
     17   11   03DF  1447          BRB     12$                             ; Send the start
                        03E1  1448
                        03E1  1449 DDCMP$STARTINT:
56   08   3C   03E1  1450          MOVZWL  S^#DLK$C_RETINFO,R6             ; Set up return to driver
     57   D4   03E4  1451          CLRL    R7                              ; Clear all set bits
     58   D5   03E6  1452          TSTL    R8                              ; If the field is zero than
     07   13   03E8  1453          BEQL    10$                             ; it was a user init start
               0434   30   03EA  1454          BSBW    DDCMP$CHK_HDR           ; Else branch to check the hdr
                 57   D5   03ED  1455          TSTL    R7                      ; If not clear than an error
                 45   12   03EF  1456          BNEQ    20$                     ; in the header.
                           03F1  1457                                          ; Continue if header ok
                 46   10   03F1  1458 10$:     BSBB    DDCMP$RESET_VAR         ; Clear msg exch variables and
          0A 0C A5    00   E0   03F3  1459          BBS     #TF$V_STATYP,TF$B_CHAR(R5),13$  ; If BS don't send STRT
                 54   06   9A   03F8  1460 12$:     MOVZBL  #DDCMP$C_STRT,R4        ; Set up a STRT to be XMT'd
               058B   30   03FB  1461          BSBW    DDCMP$QUEUE_XMT         ; Place on queue to XMT
                 57   95   03FE  1462          TSTB    R7                      ; If the reg has any bits
                           0400  1463                                          ; set than there was error
                 34   12   0400  1464          BNEQ    20$                     ; Branch to report itto driver
               0DBD   30   0402  1465 13$:     BSBW    DDCMP$RESET_GCTRS       ; Reset global counters
               0DCA   30   0405  1466          BSBW    DDCMP$RESET_TCTRS       ; Reset tributary counters
             57   04   A8   0408  1467          BISW2   #DLK$M_REINT,R7         ; Set up to inform the driver
                  02   91   040B  1468          CMPB    S^#DDCMP$C_STA_IST,-    ; Branch NEQ state was not IST
                  63         040D  1469                  GF$B_STATE(R3)          ; no need to incr threshold
                  03   12   040E  1470          BNEQ    15$
               0D8F   30   0410  1471          BSBW    DDCMP$INCR_XMT_THRS     ; Increment the threshold cntrs
          52   014C C5   DE   0413  1472 15$:     MOVAL   TF$L_TQE(R5),R2         ; Get the TQE address
20 A2  00000000 004C4B40 8F   7D   0418  1473          MOVQ    #DDCMP$C_RUNTQE,TQE$Q_DELTA(R2) ; Set the delta for active pro
          0C 0C A5    00   E1   0424  1474          BBC     #TF$V_STATYP,TF$B_CHAR(R5),18$  ; If trib mode set short delta
20 A2  00000000 002625A0 8F   7D   0429  1475          MOVQ    #DDCMP$C_TRIBTQE,TQE$Q_DELTA(R2) ; Set the delta for trib mode
                           05   0435  1476 18$:     RSB
                           0436  1477
               FDF5   31   0436  1478 20$:     BRW     DDCMP$ERROR
                           0439  1479          .DSABL  LSB
```

```
                     0439  1481
                     0439  1482          .SBTTL  Reset variables
                     0439  1483 ;RESET_VAR - Reset variables
                     0439  1484 ;
                     0439  1485 ; This routine is called on start up or reinti of the protocol to zero
                     0439  1486 ; the message exchange variables.
                     0439  1487 ;
                     0439  1488 ;       INPUTS          R3 = Address of the GF block
                     0439  1489 ;                       R5 = Address of the TF block
                     0439  1490 ;
                     0439  1491 ;       OUTPUTS         R3 and R5 are preserved.
                     0439  1492 ;
                     0439  1493 DDCMP$RESET_VAR:
                     0439  1494          ASSUME  TF$B_N EQ TF$B_R+1
                     0439  1495          ASSUME  TF$B_A EQ TF$B_N+1
                     0439  1496          ASSUME  TF$B_T EQ TF$B_A+1
                     0439  1497          ASSUME  TF$B_X EQ TF$B_T+1
                     0439  1498          ASSUME  TF$B_SELTIM EQ TF$B_X+1
                     0439  1499
          11 A5   D4 0439  1500          CLRL    TF$B_R(R5)                      ; Clear above fields
          15 A5   B4 043C  1501          CLRW    TF$B_X(R5)
        0189 C5   94 043F  1502          CLRB    TF$B_SFLAGS(R5)
        0188 C5   94 0443  1503          CLRB    TF$B_NAKRSN(R5)
          1D A5   94 0447  1504          CLRB    TF$B_MMCTR(R5)                  ; Clear then nmb of msgs sent
                     044A  1505
    20 A5   20 A5 DE 044A  1506          MOVAL   TF$Q_CTLQ(R5),TF$Q_CTLQ(R5)             ; Init control queue
    24 A5   20 A5 DE 044F  1507          MOVAL   TF$Q_CTLQ(R5),TF$Q_CTLQ+4(R5)           ; Init control queue
                     0454  1508
       50   30 B5 0F 0456  1509          REMQUE  @TF$Q_XMTQ(R5),R0
            12    1C 0458  1510          BVC     10$
       50   38 B5 0F 045A  1511          REMQUE  @TF$Q_RTOQ(R5),R0
            0C    1C 045E  1512          BVC     10$
       50   28 B5 0F 0460  1513          REMQUE  @TF$Q_CMPQ(R5),R0
            06    1C 0464  1514          BVC     10$
       50   40 B5 0F 0466  1515          REMQUE  @TF$Q_XMTOVF(R5),R0
            04    1D 046A  1516          BVS     20$
                     046C  1517 10$:     BUG_CHECK NOBUFPCKT,FATAL
                     0470  1518 ; Set up the queue block with ENQ type and subtype
                     0470  1519
10 A5    7F 8F    90 0470  1520 20$:     MOVB    #DDCMP$C_SIZEOFQ,TF$B_XQCNT(R5)         ; Set the max no of msg
                     0475  1521                                                      ; allowed on the queue
       51   5C A5 DE 0475  1522          MOVAL   TF$L_QACK(R5),R1                        ; Get ACK queue block
            0C A1 D4 0479  1523          CLRL    XMTQ$L_IRP(R1)                  ; Set no IRP address
            1F A1 94 047C  1524          CLRB    XMTQ$B_FLAG(R1)                 ; Clear flags
    1F A1    04 90 047F  1525          MOVB    #XMTQ$M_CONTROL,XMTQ$B_FLAG(R1)         ; Set cntrl flag
    1A A1    06 9B 0483  1526          MOVZBW  S^#DDCMP$C_HEADER,XMTQ$W_MSGSIZE(R1)    ; Set message size
       51    24 A1 DE 0487  1527          MOVAL   XMTQ$B_MSGADR(R1),R1                    ; Get header field
          61    05 90 048B  1528          MOVB    S^#DDCMP$C_ENQ,MFD$B_MSGID(R1)          ; Fill ENQ into msgid
    01 A1    01 9B 048E  1529          MOVZBW  #DDCMP$C_ACK,MFD$W_TYPFLG(R1)           ; Insert ACK type
                     0492  1530
 51    008C C5 DE 0492  1531          MOVAL   TF$L_QNAK(R5),R1                        ; Get NAK queue block
            0C A1 D4 0497  1532          CLRL    XMTQ$L_IRP(R1)                  ; Set no IRP address
            1F A1 94 049A  1533          CLRB    XMTQ$B_FLAG(R1)                 ; Clear flags
    1F A1    04 90 049D  1534          MOVB    #XMTQ$M_CONTROL,XMTQ$B_FLAG(R1)         ; Set cntrl flag
    1A A1    06 9B 04A1  1535          MOVZBW  S^#DDCMP$C_HEADER,XMTQ$W_MSGSIZE(R1)    ; Set message size
       51    24 A1 DE 04A5  1536          MOVAL   XMTQ$B_MSGADR(R1),R1                    ; Get header field
          61    05 90 04A9  1537          MOVB    S^#DDCMP$C_ENQ,MFD$B_MSGID(R1)          ; Fill ENQ into msgid
```

```
                01 A1    02    9B   04AC  1538          MOVZBW  #DDCMP$C_NAK,MFD$W_TYPFLG(R1)          ; Insert NAK type
                                    04B0  1539
              51    00BC C5    DE   04B0  1540          MOVAL   TF$L_QREP(R5),R1                       ; Get REP queue block
                      0C A1    D4   04B5  1541          CLRL    XMTQ$L_IRP(R1)                         ; Set no IRP address
                      1F A1    94   04B8  1542          CLRB    XMTQ$B_FLAG(R1)                        ; Clear flags
                1F A1    04    90   04BB  1543          MOVB    #XMTQ$M_CONTROL,XMTQ$B_FLAG(R1)        ; Set cntrl flag
                1A A1    06    9B   04BF  1544          MOVZBW  S^#DDCMP$C_HEADER,XMTQ$W_MSGSIZE(R1)   ; Set message size
                51    24 A1    DE   04C3  1545          MOVAL   XMTQ$B_MSGADR(R1),R1                   ; Get header field
                   61    05    90   04C7  1546          MOVB    S^#DDCMP$C_ENQ,MFD$B_MSGID(R1)         ; Fill ENQ into msgid
                01 A1    03    9B   04CA  1547          MOVZBW  #DDCMP$C_REP,MFD$W_TYPFLG(R1)          ; Insert REP type
                                    04CE  1548
              51    00EC C5    DE   04CE  1549          MOVAL   TF$L_QSTRT(R5),R1                      ; Get STRT queue block
                      0C A1    D4   04D3  1550          CLRL    XMTQ$L_IRP(R1)                         ; Set no IRP address
                      1F A1    94   04D6  1551          CLRB    XMTQ$B_FLAG(R1)                        ; Clear flags
                1F A1    04    90   04D9  1552          MOVB    #XMTQ$M_CONTROL,XMTQ$B_FLAG(R1)        ; Set cntrl flag
                1A A1    06    9B   04DD  1553          MOVZBW  S^#DDCMP$C_HEADER,XMTQ$W_MSGSIZE(R1)   ; Set message size
                51    24 A1    DE   04E1  1554          MOVAL   XMTQ$B_MSGADR(R1),R1                   ; Get header field
                   61    05    90   04E5  1555          MOVB    S^#DDCMP$C_ENQ,MFD$B_MSGID(R1)         ; Fill ENQ into msgid
                01 A1    06    9B   04E8  1556          MOVZBW  #DDCMP$C_STRT,MFD$W_TYPFLG(R1)         ; Insert STRT type
          01 A1    C000 8F    A8   04EC  1557          BISW2   #<MFD$M_QSYNC!-                        ; Set to always resync
                                    04F2  1558                  MFD$M_SELECT>,MFD$W_TYPFLG(R1)     ; and to send select
                                    04F2  1559
              51    011C C5    DE   04F2  1560          MOVAL   TF$L_QSTACK(R5),R1                     ; Get STACK queue block
                      0C A1    D4   04F7  1561          CLRL    XMTQ$L_IRP(R1)                         ; Set no IRP address
                      1F A1    94   04FA  1562          CLRB    XMTQ$B_FLAG(R1)                        ; Clear flags
                1F A1    04    90   04FD  1563          MOVB    #XMTQ$M_CONTROL,XMTQ$B_FLAG(R1)        ; Set cntrl flag
                1A A1    06    9B   0501  1564          MOVZBW  S^#DDCMP$C_HEADER,XMTQ$W_MSGSIZE(R1)   ; Set message size
                51    24 A1    DE   0505  1565          MOVAL   XMTQ$B_MSGADR(R1),R1                   ; Get header field
                   61    05    90   0509  1566          MOVB    S^#DDCMP$C_ENQ,MFD$B_MSGID(R1)         ; Fill ENQ into msgid
                01 A1    07    9B   050C  1567          MOVZBW  #DDCMP$C_STACK,MFD$W_TYPFLG(R1)        ; Insert STACK type
          01 A1    C000 8F    A8   0510  1568          BISW2   #<MFD$M_QSYNC!-                        ; Set to always resync
                                    0516  1569                  MFD$M_SELECT>,MFD$W_TYPFLG(R1)     ; and to send select
                                    05    0516  1570          RSB
```

```
                     0517  1572          .SBTTL  Start acknowledge
                     0517  1573 ;STARTACK - Start acknowledge
                     0517  1574 ;
                     0517  1575 ; This routine acknowledges starts of the protocol. If an error is found in
                     0517  1576 ; the header passed to the protocol then return immediately to the driver
                     0517  1577 ; to report a header error.
                     0517  1578 ;
                     0517  1579 ;       INPUTS:         R3 = Address of the GF block
                     0517  1580 ;                       R5 = Address of TF block
                     0517  1581 ;                       R6 = Command code from driver
                     0517  1582 ;                       R7 = Code/status associated with return states
                     0517  1583 ;                       R8 = Address of the ENQ message header
                     0517  1584 ;
                     0517  1585 ;       OUTPUTS:        R3, R5 and R8 are preserved
                     0517  1586 ;                       R6,R7 are set up with the return status for driver
                     0517  1587 ;                       XMTQ is given an entry
                     0517  1588 ;                       RTOQ is given an entry
                     0517  1589 ;
                     0517  1590 DDCMP$STARTACK:
56   08   3C         0517  1591          MOVZWL  S^#DLK$C_RETINFO,R6             ; Set up return to driver
     57   D4         051A  1592          CLRL    R7                              ; Clear all set bits
   0302   30         051C  1593          BSBW    DDCMP$CHK_HDR                   ; Chk the mdg hdr for errors
     57   D5         051F  1594          TSTL    R7                              ; Branch on NEQ to report err
     22   12         0521  1595          BNEQ    20$
                     0523  1596                                                  ; Else continue
54   07   9A         0523  1597          MOVZBL  #DDCMP$C_STACK,R4               ; Set up a STACK to be XMT'd
   0460   30         0526  1598          BSBW    DDCMP$QUEUE_XMT                 ; Place on the XMT queue
     57   95         0529  1599          TSTB    R7                              ; Branch on NEQ to report error
     18   12         052B  1600          BNEQ    20$
     03   91         052D  1601          CMPB    S^#DDCMP$C_STA_AST,-            ; Branch NEQ the state is not
     63              052F  1602                  GF$B_STATE(R3)                  ; AST do not incr the threshold
     03   12         0530  1603          BNEQ    5$
   0C6D   30         0532  1604          BSBW    DDCMP$INCR_XMT_THRS             ; Incr the threshold cntrs
     0C A5   95      0535  1605 5$:      TSTB    TF$B_CHAR(R5)                   ; If EQL then FPC station don't
     0A   13         0538  1606          BEQL    10$                             ; turn the link
     0F   E1         053A  1607          BBC     #MFD$V_SELECT,-                 ; If BC then don't turn the
  05 01 A8           053C  1608                  MFD$W_TYPFLG(R8),10$            ; link
57   4000 8F   A8    053F  1609          BISW    #DLK$M_TRNLK,R7
               05    0544  1610 10$:     RSB
                     0545  1611
         FCE6  31    0545  1612 20$:     BRW     DDCMP$ERROR
                     0548  1613
                     0548  1614
```

```
                                0548  1616
                                0548  1617          .SBTTL  Start maintenance mode
                                0548  1618 ;
                                0548  1619 ; DDCMP$STARTMNT - Start maintenance mode
                                0548  1620 ;
                                0548  1621 ; The only purpose of this routine is to provide a branch in the state
                                0548  1622 ; tables and to set the status registers for return to the driver. The
                                0548  1623 ; ulitimate goal is only to change the state of the protocol.
                                0548  1624 ;
                                0548  1625 ;       INPUTS   none
                                0548  1626 ;
                                0548  1627 ;       OUTPUTS R3, R5 and R8 are preserved
                                0548  1628 ;               R6, R7 and R9 are set to normal return
                                0548  1629 ;
                                0548  1630 DDCMP$STARTMNT:
                    56    08    3C    0548  1631          MOVZWL  S^#DLK$C_RETINFO,R6             ; Set up return to driver
                          57    D4    054B  1632          CLRL    R7                              ; Clear all set bits
                          58    7C    054D  1633          CLRQ    R8                              ; driver
                       09F1     30    054F  1634          BSBW    DDCMP$MAINT_HALT                ; Be sure to clean up the q's
                52   014C  C5    DE    0552  1635          MOVAL   TF$L_TQE(R5),R2                 ; Get the TQE address
 20 A2   00000000  004C4B40  8F    7D    0557  1636          MOVQ    #DDCMP$C_RUNTQE,TQE$Q_DELTA(R2) ; Set the delta for active pro
                                05    0563  1637          RSB
                                      0564  1638
```

```
                        0564  1640          .SBTTL  Receive message
                        0564  1641 ; DDCMP$RCVMSG - Receive message
                        0564  1642 ;
                        0564  1643 ; This routine processes received messages. It first checks the headers
                        0564  1644 ; to ensure that there is no CRC or ADDR errors. Then it goes decides
                        0564  1645 ; which type of message it is. The first thing it must do for any message
                        0564  1646 ; is to check the header fields and make sure that things are kosher.
                        0564  1647 ; For instance if it receives a data message with the NUMB field
                        0564  1648 ; less than the last message RCV'd (R) then it should ignore the message.
                        0564  1649 ; And not allocate a buffer (IRP) for it. For data messages if the message
                        0564  1650 ; header is ok then a buffer is allocated for the message. For all messages
                        0564  1651 ; the message exchange variables are updated. Only after it has determined
                        0564  1652 ; that the headers are ok will it check to see if the select has been set.
                        0564  1653 ; If it has then the selection timer is stopped and the protocol issues
                        0564  1654 ; a command to the driver that it must stop receiving messages and start
                        0564  1655 ; sending them. For H or FMT before turning the link over it must retransmit
                        0564  1656 ; all messages on the RTOQ.
                        0564  1657 ;
                        0564  1658 ; The modulo arithmetic in DDCMP is done in the following manner.
                        0564  1659 ;
                        0564  1660 ; Assume that the number of outstanding transmits will never exceed 127. This
                        0564  1661 ; allows us to use 127 as our modulus, that is we know that two message
                        0564  1662 ; exchange fields will always be between 0 and 127 away from each other. This
                        0564  1663 ; allows us to use the following formula for modulo arithmetic:
                        0564  1664 ;
                        0564  1665 ;       GIVEN:  A; B; X; MOD = 127
                        0564  1666 ;
                        0564  1667 ;               B - A --> X
                        0564  1668 ;
                        0564  1669 ;               If  X > 127 Then B < A
                        0564  1670 ;               If  X < 127 Then B > A
                        0564  1671 ;               If  X = 0   Then B = A
                        0564  1672 ;
                        0564  1673 ;       INPUTS          R3 = Address of the GF block
                        0564  1674 ;                       R5 = Address of the TF block
                        0564  1675 ;                       R8 = Address of RCV message
                        0564  1676 ;
                        0564  1677 ;       OUTPUTS         R3 and R5 are preserved
                        0564  1678 ;                       R6 and R7 are set up for return to driver
                        0564  1679 ;                       R8 is scratched
                        0564  1680 ;                       R9 is size of the data message received
                        0564  1681 ;
                        0564  1682 DDCMP$RCVMSG:
     56   08    3C      0564  1683          MOVZWL  S^#DLK$C_RETINFO,R6             ; Set up return to driver
          57    D4      0567  1684          CLRL    R7                              ; Clear all set bits
                        0569  1685
                        0569  1686          ASSUME DLK$V_CRC EQ 16
                        0569  1687
          02B5  30      0569  1688          BSBW    DDCMP$CHK_HDR                   ; Check CRC and ADDR
            57  95      056C  1689          TSTB    R7                              ; On error return to driver
            03  13      056E  1690          BEQL    2$
          0180  31      0570  1691          BRW     140$
    0C 57   10  E1      0573  1692 2$:      BBC     #DLK$V_CRC,R7,5$                ; If BC no CRC error
0188 C5     01  91      0577  1693          CMPB    #DDCMP$C_NAK1,TF$B_NAKRSN(R5)   ; If NEQ then not hdr CRC else
            05  12      057C  1694          BNEQ    5$
            58  D4      057E  1695          CLRL    R8                              ; DO not process the message
          016B  31      0580  1696          BRW     125$                            ; but start up the transmitter
```

```
                                0583  1697
                03   00   F0    0583  1698 5$:     INSV    #0,#DDCMP$C_RCV_THRS,-          ; Zero the RCV threshold
                     03         0586  1699                  #DDCMP$C_THR_SIZE,-             ; counter
                01C5 C5         0587  1700                  TF$W_THRES(R5)
                     52   D4    058A  1701          CLRL    R2                              ; If NAK rcv'd because REP
                                058C  1702                                                  ; was sent this will be set
           68   81   8F   91    058C  1703          CMPB    #DDCMP$C_SOH,MFD$B_MSGID(R8)    ; Branch if it is an SOH
                     1B   13    0590  1704          BEQL    20$
                68   05   91    0592  1705          CMPB    #DDCMP$C_ENQ,MFD$B_MSGID(R8)    ; Branch if it is an ENQ
                     13   12    0595  1706          BNEQ    10$
           51   01   A8   9A    0597  1707          MOVZBL  MFD$W_TYPFLG(R8),R1             ; Get the type of cntrl msg
                                059B  1708
                                059B  1709          $DISPATCH       R1,TYPE=B,-
                                059B  1710          <-      ;index                  ;action
                                059B  1711          <DDCMP$C_ACK                    30$>-
                                059B  1712          <DDCMP$C_NAK                    40$>-
                                059B  1713          <DDCMP$C_REP                    50$>-
                                059B  1714          >
                51   07   91    05A5  1715          CMPB    S^#DDCMP$C_STACK,R1             ; Branch if a STACK msg
                     73   13    05A8  1716          BEQL    39$
                   0143   31    05AA  1717 10$:     BRW     130$                            ; Return to driver
                                05AD  1718
                                05AD  1719 ; SOH message only
                                05AD  1720
      51   01   11   A5   81    05AD  1721 20$:     ADDB3   TF$B_R(R5),#1,R1                ; Branch on EQL because the
           51   04   A8   91    05B2  1722          CMPB    MFD$B_NUMB(R8),R1               ; incoming message is the next
                     03   13    05B6  1723          BEQL    25$                             ; message expected
                   0138   31    05B8  1724          BRW     140$                            ; Else ignore the message
                     11   A5   96    05BB  1725 25$:     INCB    TF$B_R(R5)                  ; Incr nmbr of RCV'd data msgs
                0198 C5         D6    05BE  1726          INCL    TF$L_DMSGR(R5)              ; Incr nmbr of data msgs RCV'd
             59   01   A8   3C    05C2  1727          MOVZWL  MFD$W_CNTFLG(R8),R9             ; Get the count
          59   C000 8F   AA    05C6  1728          BICW    #<MFD$M_SELECT!-                ; Clear if flags are set
                                05CB  1729                  MFD$M_QSYNC>,R9
          018C C5   59   C0    05CB  1730          ADDL2   R9,TF$L_DBYTR(R5)               ; the data msg
                     02   88    05D0  1731          BISB2   #TF$M_SACK,-                    ; Set to send an ACK
                0189 C5         05D2  1732                  TF$B_SFLAGS(R5)
                     03   91    05D5  1733          CMPB    S^#DDCMP$C_STA_AST,-            ; Branch if not AST
                     63         05D7  1734                  GF$B_STATE(R3)
                     09   12    05D8  1735          BNEQ    30$
             03   A8   00   91    05DA  1736          CMPB    #0,MFD$B_RESP(R8)               ; Branch if 0 (hdr ok)
                     30   13    05DE  1737 28$:     BEQL    35$
                   010D   31    05E0  1738          BRW     130$                            ; Report it
                                05E3  1739
                                05E3  1740 ; SOH message or an ACK received
                                05E3  1741
                                05E3  1742          ASSUME  DDCMP$C_MODULO EQ 127
                                05E3  1743
                                05E3  1744 ; Check that A < RESP <= N
                                05E3  1745
50    03   A8   13   A5   83    05E3  1746 30$:     SUBB3   TF$B_A(R5),MFD$B_RESP(R8),R0     ; Branch If RESP <= A
                     25   15    05E9  1747          BLEQ    35$                             ; no fields to update
50    03   A8   12   A5   83    05EB  1748          SUBB3   TF$B_N(R5),MFD$B_RESP(R8),R0     ; Branch if RESP > N
                     5F   14    05F1  1749          BGTR    45$                             ; To report a problem with hdr
                00   00   F0    05F3  1750          INSV    #0,#DDCMP$C_XMT_THRS,-          ; Zero the XMT threshold
                     03         05F6  1751                  #DDCMP$C_THR_SIZE -             ; counter
                01C5 C5         05F7  1752                  TF$W_THRES(R5)
          13   A5   03   A8   90    05FA  1753          MOVB    MFD$B_RESP(R8),TF$B_A(R5)       ; Update msg exchange fields
```

```
                                05FF  1754
                                05FF  1755 ; Check that   T <= A
                                05FF  1756
50   14 A5    13 A5    83      05FF  1757          SUBB3   TF$B_A(R5),TF$B_T(R5),R0        ; Branch if  T > A
                   06    14      0605  1758          BGTR    33$                          ; If GTR no field to update
        01    13 A5    81      0607  1759          ADDB3   TF$B_A(R5),#1,-              ; Update T
              14 A5            060B  1760                  TF$B_T(R5)
               08AC    30      060D  1761 33$:     BSBW    DDCMP$REM_RTOQ               ; Complete any messages ACK'd
        68    81 8F    91      0610  1762 35$:     CMPB    #DDCMP$C_SOH,MFD$B_MSGID(R8) ; If this is an SOH
                   3A    12      0614  1763          BNEQ    43$
        57   0100 8F    A8      0616  1764          BISW2   #DLK$M_RCVACK,R7              ; RCV message was ok
                   33    11      061B  1765          BRB     43$
                                061D  1766
        14 A5    01    90      061D  1767 39$:     MOVB    #1,TF$B_T(R5)                ; T must be equal to 1 at
                                0621  1768                                               ; protocol start up
                   02    88      0621  1769          BISB2   #TF$M_SACK,-                 ; Set to send an ACK
              0189 C5          0623  1770                  TF$B_SFLAGS(R5)
                   63    11      0626  1771          BRB     100$                         ; end of start up
                                0628  1772
                                0628  1773
                                0628  1774 ; NAK messages
                                0628  1775
                                0628  1776 ; Check the condition A <= RESP <= N
                                0628  1777
50   03 A8    13 A5    83      0628  1778 40$:     SUBB3   TF$B_A(R5),MFD$B_RESP(R8),R0 ; Branch RESP < A
                   22    19      062E  1779          BLSS    45$                          ; report a problem
50   03 A8    12 A5    83      0630  1780          SUBB3   TF$B_N(R5),MFD$B_RESP(R8),R0 ; Branch if RESP > N
                   1A    14      0636  1781          BGTR    45$                          ; report the problem
              00   00    F0      0638  1782          INSV    #0,#DDCMP$C_XMT_THRS,-       ; Zero the XMT threshold
                   03            063B  1783                  #DDCMP$C_THR_SIZE,-          ; counter
              01C5 C5            063C  1784                  TF$W_THRES(R5)
     13 A5    03 A8    90      063F  1785          MOVB    MFD$B_RESP(R8),TF$B_A(R5)    ; Update message fields
        01    13 A5    81      0644  1786          ADDB3   TF$B_A(R5),#1,-
              14 A5            0648  1787                  TF$B_T(R5)
               086F    30      064A  1788          BSBW    DDCMP$REM_RTOQ               ; Comp any ACK'd XMTs
               09F3    30      064D  1789          BSBW    DDCMP$LOGNAK                 ; Record why the NAK was RCV'd
                 39    11      0650  1790 43$:     BRB     100$                         ; Check select flags
               009B    31      0652  1791 45$:     BRW     130$
                                0655  1792
                                0655  1793
                                0655  1794 ; REP message
                                0655  1795
04 A8    11 A5    91      0655  1796 50$:     CMPB    TF$B_R(R5),MFD$B_NUMB(R8)    ; Ignore if R # NUMB
                   10    12      065A  1797          BNEQ    55$
                   02    88      065C  1798          BISB2   #TF$M_SACK,-               ; An ACK needs be sent
              0189 C5          065E  1799                  TF$B_SFLAGS(R5)
                   01    8A      0661  1800          BICB2   #TF$M_SNAK,-               ; Clear send NAK
              0189 C5          0663  1801                  TF$B_SFLAGS(R5)
              01C4 C5    96    0666  1802          INCB    TF$B_RRTO(R5)                ; Incr the nmbr of remote
                                066A  1803                                               ; reply timeouts
                   1F    11      066A  1804          BRB     100$                         ; Check select flags
                   03    90      066C  1805 55$:     MOVB    #DDCMP$C_NAK3,-              ; Set NAK reason
              0188 C5          066E  1806                  TF$B_NAKRSN(R5)
                   01    88      0671  1807          BISB2   #TF$M_SNAK,-               ; An NAK needs be sent
              0189 C5          0673  1808                  TF$B_SFLAGS(R5)
                   02    8A      0676  1809          BICB2   #TF$M_SACK,-               ; Clear send ACK
              0189 C5          0678  1810                  TF$B_SFLAGS(R5)
```

```
                51    01AF C5   3E  067B  1811          MOVAW   TF$B_DEI(R5),R1                 ; Incr data error inbound
                        0A53    30  0680  1812          BSBW    DDCMP$INCR_COUNTER
                          04    A8  0683  1813          BISW2   S^#TF$M_IREPS,-                 ; Set reason to be SNAK set
                      01AD C5         0685  1814                  TF$W_DEIBC(R5)              ; in repsonse to REP RCV'd
                        0AF7    30  0688  1815          BSBW    DDCMP$INCR_RCV_THRS             ; Incr the rcv threshold cnter
                       0C A5    95  068B  1816  100$:   TSTB    TF$B_CHAR(R5)                   ; If FPC ignore select flags
                          56    13  068E  1817          BEQL    120$
                          0F    E1  0690  1818          BBC     #MFD$V_SELECT,-                 ; Branch if BC else turn link
                    51 01 A8        0692  1819                  MFD$W_TYPFLG(R8),120$
                                    0695  1820  ;&      BBSS    #TF$V_OWNSELT,-                 ; Branch BS because there
                                    0695  1821  ;&              TF$B_SELTIM(R5),130$        ; is an error
                  16 A5    02    88  0695  1822          BISB2   #TF$M_OWNSELT,TF$B_SELTIM(R5)   ; Set that we own the select
                       08 A3    94  0699  1823          CLRB    GF$B_CURSEL(R3)                 ; Reset the select timeout ctr
                          03    91  069C  1824          CMPB    #DDCMP$C_NXTINTVL,-             ; Branch NEQ because not the
                       17 A5        069E  1825                  TF$B_REPTIM(R5)             ; right time to reply T/O
                          30    12  06A0  1826          BNEQ    110$
                05 A3     03 A3    B0  06A2  1827          MOVW    GF$W_SELWAI(R3),GF$W_NEXTCNT(R3) ; Set # of intvl's to deslct'n
                          52    95  06A7  1828          TSTB    R2                              ; If NEQ then NAK was rcv'd
                          27    12  06A9  1829          BNEQ    110$                            ; because a REP was sent
                          02    E1  06AB  1830          BBC     #TF$V_TIMER,-                   ; If BC then no msgs to T/O
                    22 16 A5        06AD  1831                  TF$B_SELTIM(R5),110$
                          04    88  06B0  1832          BISB2   #TF$M_SREP,-                    ; Else set REP flag to send a
                        0189 C5         06B2  1833                  TF$B_SFLAGS(R5)             ; reply to msg number
                51    01C1 C5   3E  06B5  1834          MOVAW   TF$B_LRTO(R5),R1                ; Incr the local reply timeout
                        0A19    30  06BA  1835          BSBW    DDCMP$INCR_COUNTER              ; counter
                        0AE2    30  06BD  1836          BSBW    DDCMP$INCR_XMT_THRS             ; Incr the XMT threshold cntr
                       1B A5    96  06C0  1837          INCB    TF$B_CURRTO(R5)                 ; Incr the number of times that
                                    06C3  1838                                                  ; a RTO has been snt on the msg
                       1B A5    91  06C3  1839          CMPB    TF$B_CURRTO(R5),-               ; Branch on LSS because the max
                       1A A5        06C6  1840                  TF$B_MAXRTO(R5)             ; RTO has not been reached
                          08    19  06C8  1841          BLSS    110$
                      57    02    A8  06CA  1842          BISW2   #DLK$M_PRSTERR,R7               ; Report to driver a peristant
                         1B A5    94  06CD  1843          CLRB    TF$B_CURRTO(R5)                 ; Reset the cur num of RTO's
                          21    11  06D0  1844          BRB     140$
                57    4000 8F     A8  06D2  1845  110$:   BISW2   #DLK$M_TRNLK,R7               ; Driver must know to start XMT
                          01    E0  06D7  1846          BBS     #TF$V_POINT,-                   ; Branch BS because no timer is
                    0A 0C A5        06D9  1847                  TF$B_CHAR(R5),120$          ; running for TRIB stations
                        02A6    30  06DC  1848          BSBW    DDCMP$STOP_SEL_TIMER            ; Stop selection timer
                          06    00  F0  06DF  1849          INSV    #0,#DDCMP$C_SEL_THRS,-          ; Clear the selection threshold
                          03        06E2  1850                  #DDCMP$C_THR_SIZE,-         ; counter
                        01C5 C5         06E3  1851                  TF$W_THRES(R5)
                  01 A8     07    91  06E6  1852  120$:   CMPB    S^#DDCMP$C_STACK,MFD$W_TYPFLG(R8) ; Branch if a STACK msg
                          0A    13  06EA  1853          BEQL    150$
                          58    D4  06EC  1854          CLRL    R8                              ; Clear to handle XMT
                          34    11  06EE  1855  125$:   BRB     DDCMP$XMTMSG_ALT                ; Handle sending of ENQ's and
                                    06F0  1856                                                  ; retransmits
                                    06F0  1857
                      57    08    A8  06F0  1858  130$:   BISW2   #DLK$M_HDRERR,R7               ; Fatal problem with msg RCV'd
                         FB38    31  06F3  1859  140$:   BRW     DDCMP$ERROR                     ; Return to driver
                                    06F6  1860
                          58    D4  06F6  1861  150$:   CLRL    R8                              ; Clear to handle XMT
                04 16 A5     01    E1  06F8  1862          BBC     #TF$V_OWNSELT,TF$B_SELTIM(R5),155$ ; IF BC then we don't own
                  16 A5      01    88  06FD  1863          BISB    #TF$M_SELECT,TF$B_SELTIM(R5)    ; select else send select
                        01B5    31  0701  1864  155$:   BRW     DDCMP$QEMPTY_ALT                ; Force an ACK to be XMT'd
                                    0704  1865
```

```
                        0704  1867
                        0704  1868          .SBTTL  Transmit message
                        0704  1869 ; DDCMP$XMTMSG - Transmit a message
                        0704  1870 ; DDCMP$XMTMSG_ALT
                        0704  1871 ;
                        0704  1872 ; This routine performs the transmit state table of DDCMP. It sets up
                        0704  1873 ; headers, sends NAK's, REP's and ACK's when necessary (when those
                        0704  1874 ; flags are set). It is called to retransmit data messages. Takes care of
                        0704  1875 ; the XMT message exchange variables (X,T, & N). Sending DLE msgs is
                        0704  1876 ; handled elsewhere.
                        0704  1877 ;
                        0704  1878 ; This routine starts out checking the SNAK and SREP flags. If either of these
                        0704  1879 ; are set then an NAK or REP must be sent. If these are clear it checks
                        0704  1880 ; to see if any messages need be retransmitted. It retransmits all messages
                        0704  1881 ; required for retransmit before it sends any new data. New data is queued to
                        0704  1882 ; the overflow queue first. This means that if any messages are outstanding
                        0704  1883 ; the data will not get sent until that condition is resolved. If none of
                        0704  1884 ; these conditions are true the messages are taken from the front of the
                        0704  1885 ; XMTOVF queue and given to the XMTQ until it is full. ACKs are piggybacked
                        0704  1886 ; on other messages except REPs when possible else are sent when the XMTer
                        0704  1887 ; has no other work.
                        0704  1888 ;
                        0704  1889 ;       INPUTS          R3 = Address of the GF block
                        0704  1890 ;                       R5 = Address of the TF block
                        0704  1891 ;                       R8 = Address of data for SOH message
                        0704  1892 ;
                        0704  1893 ;       OUTPUTS         R3, R5 and R8 are preserved
                        0704  1894 ;                       R6 and R7 contain return information for driver
                        0704  1895 ;                       Messages to be sent are placed on the XMTQ and RTOQ
                        0704  1896 ;
                        0704  1897          .ENABL LSB
                        0704  1898 DDCMP$XMTMSG:
     56    08    3C     0704  1899          MOVZWL  S^#DLK$C_RETINFO,R6             ; Set up return to driver
           57    D4     0707  1900          CLRL    R7                              ; Clear all set bits
           58    D5     0709  1901          TSTL    R8                              ; If Eql then no data to XMT
           17    13     070B  1902          BEQL    DDCMP$XMTMSG_ALT
     1A    A8    B5     070D  1903          TSTW    XMTQ$W_MSGSIZE(R8)              ; If NEQ then legal size
           03    12     0710  1904          BNEQ    5$
        00FB     31     0712  1905          BRW     DDCMP$XMTERR
  44    B5    68     0E  0715  1906 5$:     INSQUE  (R8),@TF$Q_XMTOVF+4(R5)         ; Assume msg can't be sent
  13     0A   A8     91  0719  1907          CMPB    XMTQ$B_BUFTYP(R8),S^#DYN$C_BUFIO ; If EQL then correct buf
           34    12     071D  1908          BNEQ    14$
57    0200    8F     A8  071F  1909          BISW2   #DLK$M_XMTACK,R7                 ; Msg to XMT has been queue'd
                        0724  1910
                        0724  1911 DDCMP$XMTMSG_ALT:                              ; When coming from RCVMSG
           59    DD     0724  1912          PUSHL   R9                              ; Must save size from RCVMSG
           00    E1     0726  1913 8$:      BBC     #TF$V_SNAK,-                    ; If not BC send NAK
  0F 0189    C5         0728  1914                  TF$B_SFLAGS(R5),10$
     54    02    9A     072C  1915          MOVZBL  #DDCMP$C_NAK,R4                 ; Set up that it is a NAK to Q
        0257    30     072F  1916          BSBW    DDCMP$QUEUE_XMT                 ; Q the NAK, hdr is also setup
           57    95     0732  1917          TSTB    R7                              ; Branch on NEQ to report error
           21    12     0734  1918          BNEQ    15$
0189    C5    03    8A  0736  1919          BICB2   #<-
                        073B  1920                  TF$M_SNAK!-                     ; Clear send NAK flag
                        073B  1921                  TF$M_SACK-                      ; Clear send ACK flag
                        073B  1922                  >,TF$B_SFLAGS(R5)
           02    E1     073B  1923 10$:     BBC     #TF$V_SREP,-                    ; If not BC send REP
```

```
                19 0189 C5          073D  1924                BBS     TF$B_SFLAGS(R5),20$
                   54    03    9A   0741  1925                MOVZBL  #DDCMP$C_REP,R4          ; Set up that it is a REP to Q
                         0242  30   0744  1926                BSBW    DDCMP$QUEUE_XMT         ; Q the REP, hdr is also setup
                            57 95   0747  1927                TSTB    R7                      ; Branch on NEQ to report error
                            0C 12   0749  1928                BNEQ    15$
                            04 8A   074B  1929                BICB2   #TF$M_SREP,-            ; Clear send REP flag (REP has
                      0189 C5       074D  1930                        TF$B_SFLAGS(R5)         ; been sent
                                    0750  1931
                         00B3 31    0750  1932 12$:     BRW     60$                     ; If a reply is sent don't XMT
                                    0753  1933                                        ; more messages
                                    0753  1934 14$:     BUG_CHECK       NOBUFPCKT,FATAL
                         00B0 31    0757  1935 15$:     BRW     70$                     ; Return in error
                                    075A  1936
                   51    12 A5 90   075A  1937 20$:     MOVB    TF$B_N(R5),R1           ; Any messages to reXMT
                50 14 A5    51 83   075E  1938          SUBB3   R1,TF$B_T(R5),R0        ; Branch  N  <= T
                            07 14   0763  1939          BGTR    25$                     ; No msgs to retransmit
                         036E 30    0765  1940          BSBW    DDCMP$REQUEUE_XMT       ; If LEQ then req MSG(T)
                            57 95   0768  1941          TSTB    R7                      ; If NEQ then prst error found
                            EB 12   076A  1942          BNEQ    15$
                                    076C  1943
                                    076C  1944 ; The following check is not necessary in full duplex operation.
                                    076C  1945
                         0C A5 95   076C  1946 25$:     TSTB    TF$B_CHAR(R5)           ; If FPC then ignore the check
                            05 13   076F  1947          BEQL    30$
                                    0771  1948
                                    0771  1949 ; The following code is here to ensure that we don't queue a message on the
                                    0771  1950 ; XMTQ until we own the select flag. This is done so that we can ensure
                                    0771  1951 ; that the latest RESP data is send out in the next selection interval. It
                                    0771  1952 ; was added in to cure the problem of a high number of Remote reply timeouts
                                    0771  1953 ; on the DMF32 when running in half duplex and trib modes.
                                    0771  1954
                DA 16 A5    01  E1  0771  1955          BBC     #TF$V_OWNSELT,TF$B_SELTIM(R5),12$
                                    0776  1956
                                    0776  1957 ; Now T = N + 1 and SREP and SNAK flags are clear, send SOH messages
                                    0776  1958
                   58    40 B5 0F   0776  1959 30$:     REMQUE  @TF$Q_XMTOVF(R5),R8     ; Get the next message to XMT
                            D4 1D   077A  1960          BVS     12$                     ; If VS then queue empty
                            0F 13   077C  1961          BEQL    33$                     ; If NEQ then more to come
                         0C A5 95   077E  1962          TSTB    TF$B_CHAR(R5)           ; If this is not a FPC sta
                            11 13   0781  1963          BEQL    34$
                         1D A5 96   0783  1964          INCB    TF$B_MMCTR(R5)          ; Incr the max msg counter
                1D A5    1C A5 91   0786  1965          CMPB    TF$B_MSGCNT(R5),TF$B_MMCTR(R5) ; If LSS then max amount of
                            07 14   078B  1966          BGTR    34$                     ; msg to xmt in an interval
                            01 88   078D  1967 33$:     BISB2   #TF$M_SELECT,-          ; Set to send a select flag
                         16 A5      078F  1968                  TF$B_SELTIM(R5)
                         1D A5 94   0791  1969          CLRB    TF$B_MMCTR(R5)
                            5A DD   0794  1970 34$:     PUSHL   R10                     ; Save a register
                   5A    16 A5 90   0796  1971          MOVB    TF$B_SELTIM(R5),R10      ; Pick up select flag
                   13    0A A8 91   079A  1972          CMPB    XMTQ$B_BUFTYP(R8),S^#DYN$C_BUFIO ; If EQL then correct buf
                            B3 12   079E  1973          BNEQ    14$
                            06 A3   07A0  1974 35$:     SUBW3   #MFD$K_LENGTH,-         ; Set size of data message
                   59    1A A8      07A2  1975                  XMTQ$W_MSGSIZE(R8),R9
                            51 D4   07A5  1976          CLRL    R1                      ; Clear the reg of old addr
                   54    81 8F 9A   07A7  1977          MOVZBL  #DDCMP$C_SOH,R4         ; SOH header to setup
                         01DB 30    07AB  1978          BSBW    DDCMP$QUEUE_XMT         ; Queue the msg also hdr setuP
                51 57       05 E0   07AE  1979          BBS     #DLK$V_QFULERR,R7,50$   ; Message not on XMTQ
                         12 A5 96   07B2  1980          INCB    TF$B_N(R5)              ; Increment the highest XMT'd
```

```
                12 A5    81   07B5  1981          ADDB3   TF$B_N(R5),-                    ; Set next to XMT
              14 A5 01          07B8  1982                  #1,TF$B_T(R5)
                 0C A5    95   07BB  1983          TSTB    TF$B_CHAR(R5)                   ; If this is not a FPC sta
                    43    13   07BE  1984          BEQL    50$
                          07C0  1985
              3F 5A 00    E1   07C0  1986          BBC     #TF$V_SELECT,R10,50$            ; BR if did not send select flg
                    02    91   07C4  1987          CMPB    #DDCMP$C_EGINTVL,-              ; Branch NEQ the station is
                 17 A5          07C6  1988                  TF$B_REPTIM(R5)                 ; Multipoint control station
                    39    12   07C8  1989          BNEQ    50$                             ; reply T/O must be handled
                    02    E1   07CA  1990          BBC     #TF$V_TIMER,-                   ; If BC then timer not running
                34 16 A5          07CC  1991                  TF$B_SELTIM(R5),50$             ; no need to send a REP
                    04    88   07CF  1992          BISB2   #TF$M_SREP,-                    ; Else must send a REP before
                 0189 C5          07D1  1993                  TF$B_SFLAGS(R5)                 ; the msg with the select flag
           51    01C1 C5    3E   07D4  1994          MOVAW   TF$B_LRTO(R5),R1                ; Incr the local reply timeout
                   08FA     30   07D9  1995          BSBW    DDCMP$INCR_COUNTER              ; error counter
                   09C3     30   07DC  1996          BSBW    DDCMP$INCR_XMT_THRS             ; Incr the XMT threshold cntr
                 1B A5      96   07DF  1997          INCB    TF$B_CURRTO(R5)                 ; Incr the number of times that
                          07E2  1998                                                        ; a RTO has been snt on the msg
                 1B A5      91   07E2  1999          CMPB    TF$B_CURRTO(R5),-               ; Branch on LSS because the max
                 1A A5          07E5  2000                  TF$B_MAXRTO(R5)                 ; RTO has not been reached
                    0B      19   07E7  2001          BLSS    40$
                 1B A5      94   07E9  2002          CLRB    TF$B_CURRTO(R5)                 ; Reset the cur num of RTO's
                 57 02      A8   07EC  2003          BISW2   #DLK$M_PRSTERR,R7               ; Report to driver a peristant
                    5A 8ED0  07EF  2004          POPL    R10                             ; Restore R10
                    16      11   07F2  2005          BRB     70$
                 54 03      9A   07F4  2006 40$:     MOVZBL  #DDCMP$C_REP,R4                 ; Set up that it is a REP to Q
                   018F     30   07F7  2007          BSBW    DDCMP$QUEUE_XMT                 ; Q the REP, hdr is also setup
                    57      95   07FA  2008          TSTB    R7                              ; Branch on NEQ to report error
                    05      12   07FC  2009          BNEQ    50$
                    04      8A   07FE  2010          BICB2   #TF$M_SREP,-                    ; Clear send REP flag (REP has
                 0189 C5          0800  2011                  TF$B_SFLAGS(R5)                 ; been sent
                    5A 8ED0  0803  2012 50$:     POPL    R10                             ; Restore R10
                    59 8ED0  0806  2013 60$:     POPL    R9
                          05   0809  2014          RSB
                          080A  2015
                    59 8ED0  080A  2016 70$:     POPL    R9
                   FA1E     31   080D  2017          BRW     DDCMP$ERROR
                          0810  2018
                          0810  2019          .DSABL LSB
                          0810  2020
                          0810  2021 DDCMP$XMTERR:
                 57 10      A8   0810  2022          BISW    #DLK$M_XMTERR,R7                ; Set error on XMT
           51    23 A3      9E   0813  2023          MOVAB   GF$B_LSE(R3),R1                 ; Bump error counters
                   08BC     30   0817  2024          BSBW    DDCMP$INCR_COUNTER
                    04      A8   081A  2025          BISW2   #GF$M_LXMT_UNDER,-              ; XMT underruns
                 21 A3          081C  2026                  GF$W_LSE_BCTRS(R3)
                   FA0D     31   081E  2027          BRW     DDCMP$ERROR
                          0821  2028
```

```
                                0821  2030
                                0821  2031          .SBTTL  Check the receive message header
                                0821  2032 ;DDCMP$CHK_HDR - Check the receive message header
                                0821  2033 ;
                                0821  2034 ; This routine checks the receive messages' header to ensure that the header
                                0821  2035 ; has no errors. It also checks the CRC's if the device this protocol is
                                0821  2036 ; running on does not support CRC checking.
                                0821  2037 ;
                                0821  2038 ;       INPUTS          R3 = Address of the GF block
                                0821  2039 ;                       R5 = Address of the TF block
                                0821  2040 ;                       R7 = Clear
                                0821  2041 ;                       R8 = Address of message header
                                0821  2042 ;
                                0821  2043 ;       OUTPUTS         R7 = Bits are set according to the errors found
                                0821  2044 ;                       R3, R5,and R8 are preserved
                                0821  2045 ;
                                0821  2046 DDCMP$CHK_HDR:
          4C 02 A3    00    E1  0821  2047          BBC     #GF$V_CRC,GF$B_DRVCHR(R3),10$   ; If not BC then chk CRC's
                              1C  BB  0826  2048          PUSHR   #^M<R2,R3,R4>
    68    06    00    F858  CF  0B  0828  2049          CRC     CRCTABLE,#0,#MFD$K_LENGTH,(R8)  ; Calc the header CRC
                              1C  BA  082F  2050          POPR    #^M<R2,R3,R4>
            51    06    A8  B0  0831  2051          MOVW    MFD$K_LENGTH(R8),R1             ; Get CRC from Message
                  50    51  B1  0835  2052          CMPW    R1,R0                           ; If EQL then no lost data
                              07  13  0838  2053          BEQL    5$
              0188  C5    01  90  083A  2054          MOVB    #DDCMP$C_NAK1,TF$B_NAKRSN(R5)   ; Set the reason for NAK
                              42  11  083F  2055          BRB     20$                             ; Branch to set NAK flag
              68    81    8F  91  0841  2056 5$:      CMPB    #DDCMP$C_SOH,MFD$B_MSGID(R8)    ; If not a data message CRC
                              2B  12  0845  2057          BNEQ    10$                             ; is done
            51    01    A8  B0  0847  2058          MOVW    MFD$W_CNTFLG(R8),R1             ; Get the count field
          51    C000  8F    AA  084B  2059          BICW    #<MFD$M_SELECT!-                ; Clear if flags are set
                                0850  2060                  MFD$M_QSYNC>,R1
          59    51    08    C1  0850  2061          ADDL3   #MFD$K_LENGTH+2,R1,R9           ; Get size of hdr hcrc and data
                              1C  BB  0854  2062          PUSHR   #^M<R2,R3,R4>
08 A8     51    00    F82A  CF  0B  0856  2063          CRC     CRCTABLE,#0,R1,-                ; Calc CRC for data
                                085E  2064                  MFD$K_LENGTH+2(R8)
                              1C  BA  085E  2065          POPR    #^M<R2,R3,R4>
                  51    58  D0  0860  2066          MOVL    R8,R1                           ; Get RCV message address
                  51    59  C0  0863  2067          ADDL2   R9,R1                           ; Add hdr, hcrc and data to get dcrc
                  61    50  B1  0866  2068          CMPW    R0,(R1)                         ; Branch EQL the data CRC was
                              07  13  0869  2069          BEQL    10$                             ; was OK
              0188  C5    02  90  086B  2070          MOVB    #DDCMP$C_NAK2,TF$B_NAKRSN(R5)   ; Set NAK reason and
                              11  11  0870  2071          BRB     20$                             ; Branch to set NAk flag
                    0C    A5  95  0872  2072 10$:     TSTB    TF$B_CHAR(R5)                   ; If full point control skip
                              1B  13  0875  2073          BEQL    30$                             ; Address checking
                    05    A8  91  0877  2074          CMPB    MFD$B_ADDR(R8),-                ; Do the addr match
                    0D    A5      087A  2075                  TF$B_ADDR(R5)
                              14  13  087C  2076          BEQL    30$                             ; Yes
                              11  90  087E  2077          MOVB    #DDCMP$C_NAK17,-                ; Set NAK rsn
                  0188  C5        0880  2078                  TF$B_NAKRSN(R5)
                              01  88  0883  2079 20$:     BISB2   #TF$M_SNAK,-                    ; Set to send the NAK
                  0189  C5        0885  2080                  TF$B_SFLAGS(R5)
        57    00010000  8F    C8  0888  2081          BISL2   #DLK$M_CRC,R7                   ; Report the error to driver
                    08F0    30  088F  2082          BSBW    DDCMP$INCR_RCV_THRS             ; Incr the receive threshold
                                0892  2083                                                  ; counter
                              05  0892  2084 30$:     RSB
                                0893  2085
```

```
                     0893  2087
                     0893  2088          .SBTTL  Reply timing
                     0893  2089 ;DDCMP$REPTIM - Reply Timing
                     0893  2090 ;
                     0893  2091 ; This routine is called after a message has been transmitted by the driver.
                     0893  2092 ; When the routine receives a message that must have reply timeout it enters
                     0893  2093 ; it onto the RTOQ. In the case where the message transmitted
                     0893  2094 ; does need to have a reply timeout the routine returns a normal status to the
                     0893  2095 ; driver.
                     0893  2096 ;
                     0893  2097 ;       INPUTS          R5 = Address of the TF block
                     0893  2098 ;                       R8 = Address of the XMT'd message
                     0893  2099 ;
                     0893  2100 ;       OUTPUTS         R6,R7 contain the return status for driver
                     0893  2101 ;                       R3, R5, and R8 are preserved
                     0893  2102 ;                       R9 is cleared
                     0893  2103 ;
                     0893  2104 DDCMP$REPTIM:
          56    08   3C   0893  2105          MOVZWL  S^#DLK$C_RETINFO,R6             ; Set up return status for
                57   D4   0896  2106          CLRL    R7                              ; driver
        51    24 A8   DE   0898  2107          MOVAL   XMTQ$B_MSGHDR(R8),R1            ; Get the addr of the msg hdr
        61    81 8F   91   089C  2108          CMPB    #DDCMP$C_SOH,MFD$B_MSGID(R1)    ; Branch on NEQ not SOH msg
                 0B   12   08A0  2109          BNEQ    20$
               05CA   30   08A2  2110          BSBW    DDCMP$INS_RTOQ                  ; Insert the message on RTOQ
     15 A5     04 A1   90   08A5  2111          MOVB    MFD$B_NUMB(R1),TF$B_X(R5)       ; Set X = NUMB
               060F   31   08AA  2112          BRW     DDCMP$REM_RTOQ                  ; Complete XMT's on RTOQ ACK'd
                      05   08AD  2113 20$:     RSB
                           08AE  2114
```

```
                        08AE  2116
                        08AE  2117          .SBTTL  Queue empty - protocol may have work
                        08AE  2118
                        08AE  2119 ;DDCMP$QEMPTY - Queue  empty
                        08AE  2120 ;
                        08AE  2121 ; This routine is called when the user detects that the XMTQ is empty
                        08AE  2122 ; but the protocol may have ACK's to send.
                        08AE  2123 ;
                        08AE  2124 ;         INPUTS          R3 = Address of the GF block
                        08AE  2125 ;                         R5 = Address of the TF block
                        08AE  2126 ;
                        08AE  2127 ;         OUTPUTS         R3 and R5 are preserved
                        08AE  2128 ;                         R6,R7 contain status info
                        08AE  2129 ;                         Other registers are destroyed
                        08AE  2130 ;
                        08AE  2131          .ENABL LSB
                        08AE  2132 DDCMP$QEMPTY:
            56   02  3C  08AE  2133          MOVZWL  S^#DLK$C_XMTMSG,R6              ; Set up return status for
                 57  D4  08B1  2134          CLRL    R7                              ; Set up for return status
      0D 0189 C5   01  E5  08B3  2135          BBCC    #TF$V_SACK,TF$B_SFLAGS(R5),20$  ; If BC then no ACK to send
                        08B9  2136
                        08B9  2137 DDCMP$QEMPTY_ALT:
            54   01  9A  08B9  2138 10$:      MOVZBL  S^#DDCMP$C_ACK,R4               ; Set to send and ACK
                00CA  30  08BC  2139          BSBW    DDCMP$QUEUE_XMT                 ; Queue the ACK
                 57  95  08BF  2140          TSTB    R7                              ; If EQL then error msg not
                 03  12  08C1  2141          BNEQ    20$                             ; sent
            57   01  A8  08C3  2142          BISW    #DLK$M_MSGSENT,R7               ; Set that the ACK is on q
                     05  08C6  2143 20$:      RSB
                        08C7  2144          .DSABL LSB
                        08C7  2145
```

```
                     08C7  2147
                     08C7  2148          .SBTTL  Selection timeout clock
                     08C7  2149
                     08C7  2150 ; START_TIMER - Selection timeout clock
                     08C7  2151 ;
                     08C7  2152 ; This routine sets the time the selection timer should expire, if it hasn't
                     08C7  2153 ; received some acknowmledgement from the other side.
                     08C7  2154 ;
                     08C7  2155 ;         INPUTS          R3 = GF block address
                     08C7  2156 ;
                     08C7  2157 ;         OUTPUTS         R3 is preserved
                     08C7  2158 ;
                     08C7  2159 ;
                     08C7  2160 ;
                     08C7  2161 DDCMP$START_SEL_TIMER:
50   00000000'GF   D0  08C7  2162          MOVL    G^EXE$GL_ABSTIM,R0              ; Get the current time
     51   03 A3    3C  08CE  2163          MOVZWL  GF$W_SELOAI(R3),R1              ; Get selection time wait
 0D A3    51   50   C1  08D2  2164          ADDL3   R0,R1,GF$L_SELEND(R3)           ; Add the wait time to current
                    05  08D7  2165          RSB                                     ; time to get select end time
                     08D8  2166
```

```
                                08D8  2168
                                08D8  2169 ;DDCMP$START_TIMER
                                08D8  2170 ;
                                08D8  2171 ; Starts the TQE running must be called before the protocol is started
                                08D8  2172 ;
                                08D8  2173 ;       INPUTS          R3 = Address of the GF block
                                08D8  2174 ;                       R5 = Address of TF block
                                08D8  2175 ;                       R7 = Address of the driver timer routine
                                08D8  2176 ;                       R8 = UCB address for device
                                08D8  2177 ;                       R9 = Cleared on input
                                08D8  2178 ;
                                08D8  2179 ;       IPL must be a device fork IPL
                                08D8  2180 ;
                                08D8  2181 ;       OUTPUTS         R3 and R5 are preserved
                                08D8  2182 ;                       R6 & R7 return status
                                08D8  2183 ;
                                08D8  2184 DDCMP$START_TIMER:
                   56    08   3C  08D8  2185          MOVZWL  S^#DLK$C_RETINFO,R6             ; Set up return info
                50 A5    57   D0  08DB  2186          MOVL    R7,TF$A_DEV_TIMER(R5)           ; Set up device timer rou addr
                         57   D4  08DF  2187          CLRL    R7                              ; Clear for return sts
                      00E8 8F   BB  08E1  2188          PUSHR   #^M<R3,R5,R6,R7>                ; Save registers
                         54   55   D0  08E5  2189          MOVL    R5,R4                           ; Set up to save R4
                55    014C C5   DE  08E8  2190          MOVAL   TF$L_TQE(R5),R5                 ; Get the address of the TQE
                  10 A5    58   D0  08ED  2191          MOVL    R8,TQE$L_FR3(R5)                ; Save UCB address
                  14 A5    54   D0  08F1  2192          MOVL    R4,TQE$L_FR4(R5)                ; Save TFB address
          0C A5    00000CE1'GF   DE  08F5  2193          MOVAL   G^DDCMP$TIMER,TQE$L_FPC(R5)     ; Set address of timer routine
                   0B A5    05   90  08FD  2194          MOVB    #TQE$C_SSREPT,TQE$B_RQTYPE(R5)  ; Set repeat bit sets
                                  0901  2195                                                  ; TF$M_REPEAT also
  20 A5    00000000 01C9C380 8F   7D  0901  2196          MOVQ    #DDCMP$C_STPTQE,TQE$Q_DELTA(R5) ; Set delta for halted protocol
                                  090D  2197
                      11 A3    D4  090D  2198          CLRL    GF$L_TQE_STS(R3)                ; Clear status
                50    00000000'GF   7D  0910  2199          MOVQ    G^EXE$GQ_SYSTIME,R0             ; Get system time
                      28 A5    94  0917  2200          CLRB    TQE$B_RMOD(R5)                  ; Set no access mode for TQE
                00000000'GF    16  091A  2201          JSB     G^EXE$INSTIMQ                   ; Insert the TQE on timer queue
                                  0920  2202
                      00E8 8F   BA  0920  2203          POPR    #^M<R3,R5,R6,R7>                        ; Restore registers
             30 A5    30 A5    DE  0924  2204          MOVAL   TF$Q_XMTQ(R5),TF$Q_XMTQ(R5)             ; Init transmit queue
             34 A5    30 A5    DE  0929  2205          MOVAL   TF$Q_XMTQ(R5),TF$Q_XMTQ+4(R5)           ; Init transmit queue
             20 A5    20 A5    DE  092E  2206          MOVAL   TF$Q_CTLQ(R5),TF$Q_CTLQ(R5)             ; Init control queue
             24 A5    20 A5    DE  0933  2207          MOVAL   TF$Q_CTLQ(R5),TF$Q_CTLQ+4(R5)           ; Init control queue
             38 A5    38 A5    DE  0938  2208          MOVAL   TF$Q_RTOQ(R5),TF$Q_RTOQ(R5)             ; Init reply timeout Q
             3C A5    38 A5    DE  093D  2209          MOVAL   TF$Q_RTOQ(R5),TF$Q_RTOQ+4(R5)           ; Init reply timeout Q
             28 A5    28 A5    DE  0942  2210          MOVAL   TF$Q_CMPQ(R5),TF$Q_CMPQ(R5)             ; Init completetion Q
             2C A5    28 A5    DE  0947  2211          MOVAL   TF$Q_CMPQ(R5),TF$Q_CMPQ+4(R5)           ; Init completetion Q
             40 A5    40 A5    DE  094C  2212          MOVAL   TF$Q_XMTOVF(R5),TF$Q_XMTOVF(R5)         ; Init overflow queue
             44 A5    40 A5    DE  0951  2213          MOVAL   TF$Q_XMTOVF(R5),TF$Q_XMTOVF+4(R5)
                01 A3    01   88  0956  2214          BISB2   #GF$M_TIMER_RUNNING,GF$B_TIMER_STATE(R3) ; Set that timer is on
                                  095A  2215          RSB
                                  095B  2216
```

```
                        095B  2218
                        095B  2219 ;DDCMP$STOP_TIMER
                        095B  2220 ;
                        095B  2221 ; Stops the TQE running must be called after the protocol is halted
                        095B  2222 ;
                        095B  2223 ;       INPUTS          R3 = Address of the GF block
                        095B  2224 ;                       R5 = Address of TF block
                        095B  2225 ;
                        095B  2226 ;       IPL must be a device fork IPL
                        095B  2227 ;
                        095B  2228 ;       OUTPUTS         R3 and R5 are preserved
                        095B  2229 ;                       R6 & R7 return status
                        095B  2230 ;
                        095B  2231 DDCMP$STOP_TIMER::
          56   08   3C  095B  2232         MOVZWL  S^#DLK$C_RETINFO,R6             ; Set up return info
               57   D4  095E  2233         CLRL    R7                              ; Clear for return sts
          00E8 8F   BB  0960  2234         PUSHR   #^M<R3,R5,R6,R7>
               52   D4  0964  2235         CLRL    R2                              ; Set no access mode of TQE
                        0966  2236 ;
                        0966  2237 ; The following set up for the call to remove the timer queue element may
                        0966  2238 ; look strange. Each of these fields has been set up this way so that
                        0966  2239 ; first only one TQE (the one we are interested in removing) is removed
                        0966  2240 ; from the timer queue, second the routine must be faked out so that
                        0966  2241 ; it doesn't try to deallocate this space from nonpaged pool (thats why
                        0966  2242 ; we said it was SSSNGL when the TQE is in practice repeatable).
                        0966  2243 ;
     50   014C C5   DE  0966  2244         MOVAL   TF$L_TQE(R5),R0                 ; Set up TQE address
       55   14 A0   D0  096B  2245         MOVL    TQE$L_FR4(R0),R5                ; Get the TF block address
        0C A0   55   D0  096F  2246         MOVL    R5,TQE$L_FPC(R0)                ; Set up in FPC so rigth TQE is  dea
               54   01  9A  0973  2247         MOVZBL  #TQE$C_SSSNGL,R4                ; Set system proc not repeatable
     00000000'GF   16  0976  2248         JSB     G^EXE$RMVTIMQ                   ; Call system routine to rmv TQE
          00E8 8F   BA  097C  2249         POPR    #^M<R3,R5,R6,R7>
        01 A3   01   8A  0980  2250         BICB2   #GF$M_TIMER_RUNNING,GF$B_TIMER_STATE(R3) ; Set that timer is off
                        05  0984  2251         RSB
                        0985  2252
```

```
                     0985  2254 ;DDCMP$STOP_SEL_TIMER
                     0985  2255 ;
                     0985  2256 ; This routine stops the selection timer by clearing the bit to say nothing is
                     0985  2257 ; being timed.
                     0985  2258 ;
                     0985  2259 ;       INPUTS          R5 = Address of the TF block
                     0985  2260 ;
                     0985  2261 ;
                     0985  2262 DDCMP$STOP_SEL_TIMER:
1D A5          94    0985  2263         CLRB    TF$B_MMCTR(R5)                  ; Clear the number of msgs XMTd
               05    0988  2264         RSB
                     0989  2265
                     0989  2266
```

```
                0989  2268
                0989  2269          .SBTTL  Queue  and requeue the message to be transmitted
                0989  2270 ; QUEUE_XMT - Queue transmission
                0989  2271 ;
                0989  2272 ; This routine is responsible for the XMT Queue (XMTQ). When a mesage is
                0989  2273 ; XMT'd this rcutine is called to manage the queue. If it is a DLE message
                0989  2274 ; then it is only queued when XMTQ is empty. SOH and ENQ's are transmitted
                0989  2275 ; in the following order:
                0989  2276 ;                   NAK,REP,(N * SOH),ACK
                0989  2277 ; A NAK and ACK should not be on the queue simultaneously.
                0989  2278 ; This routine also calls SETUP_HDR to include the header message in the
                0989  2279 ; queue, hence when the driver is to transmit the message it must first
                0989  2280 ; set up its' header.
                0989  2281 ;
                0989  2282 ; XMTQ is 'ull when each of the slots allowed to transmit data messages is
                0989  2283 ; filled or inthe case of retransmitting when that value is less than or
                0989  2284 ; equal to zero. In such cases an error is returned and the transmit is
                0989  2285 ; queued to the overflow queue.
                0989  2286 ;
                0989  2287 ; NOTE: Any change to the XMTQ or control messages on that queue must be
                0989  2288 ;       done at device IPL
                0989  2289 ;
                0989  2290 ;       INPUTS          R3 = Address of the GF block
                0989  2291 ;                       R4 = Type of message
                0989  2292 ;                       R5 = address of TF block
                0989  2293 ;                       R8 = Address of buffer to queue for data and maint
                0989  2294 ;                       messages only
                0989  2295 ;                       R9 = Size of data message
                0989  2296 ;
                0989  2297 ;       OUTPUTS         R3, R5, R8, and R9 are preserved
                0989  2298 ;
                0989  2299 ;
                0989  2300 ;       A new entry is put onto the XMT QUEUE
                0989  2301 ;
                0989  2302 DDCMP$QUEUE_XMT:
                0989  2303
                0989  2304 ; Handle SOH message
                0989  2305
54   81 8F   91 0989  2306          CMPB    #DDCMP$C_SOH,R4                 ; Branch if not SOH
     39   12 098D  2307          BNEQ    20$
     10 A5   95 098F  2308          TSTB    TF$B_XQCNT(R5)                  ; Br NEQ XMTQ  is not yet
     06   12 0992  2309          BNEQ    5$                              ; filled
57   20   A8 0994  2310          BISW2   #DLK$M_QFULERR,R7               ; Set queue full error
   012D   31 0997  2311 2$:      BRW     150$
     10 A5   97 099A  2312 5$:      DECB    TF$B_XQCNT(R5)                  ; If free grab it
52   58   D0 099D  2313          MOVL    R8,R2                           ; Save in R2 for CRC calc
   022F   30 09A0  2314          BSBW    DDCMP$SETUP_MSG_HDR             ; Set up the msg header
          09A3  2315          DSBINT  GF$B_DIPL(R3)                   ; Disable device interrupts
13   0A A8   91 09AA  2316          CMPB    XMTQ$B_BUFTYP(R8),S^#DYN$C_BUFIO ; If EQL then correct buf
     04   13 09AE  2317          BEQL    10$
          09B0  2318          BUG_CHECK       NOBUFPCKT,FATAL
34 B5   68   0E 09B4  2319 10$:    INSQUE  (R8),@TF$Q_XMTQ+4(R5)           ; Q at end of XMTQ
     0C A5   95 09B8  2320          TSTB    TF$B_CHAR(R5)                   ; If EQL then don't start
     08   13 09BB  2321          BEQL    15$                             ; selection timer
     0F   E1 09BD  2322          BBC     #MFD$V_SELECT,-                 ; If BC no select to process
  03 01 A1       09BF  2323                  MFD$W_CNTFLG(R1),15$
     00C4   31 09C2  2324          BRW     100$                            ; Branch to start timer
```

```
                00FB   31  09C5  2325 15$:    BRW     120$                            ; Branch to comp queueing
                       09C8  2326
                       09C8  2327 ; Handle DLE message
                       09C8  2328
          54    90 8F  91  09C8  2329 20$:    CMPB    #DDCMP$C_DLE,R4                 ; Branch if not DLE
                   1B  12  09CC  2330          BNEQ    30$
              52   58  D0  09CE  2331          MOVL    R8,R2                           ; Save in R2 for CRC calc
                 01FE  30  09D1  2332          BSBW    DDCMP$SETUP_MSG_HDR             ; Set up the header
                       09D4  2333          DSBINT  GF$B_DIPL(R3)                   ; Disable device interrupt
          30 A5    68  0E  09DB  2334          INSQUE  (R8),TF$Q_XMTQ(R5)              ; Queue the msg to XMT
              10 A5    97  09DF  2335          DECB    TF$B_XQCNT(R5)                  ; Dec number of free places
          1F A8    02  A8  09E2  2336          BISW2   #XMTQ$M_SELECT,XMTQ$B_FLAG(R8)  ; Driver must know to turn link
                 00DA  31  09E6  2337 25$:    BRW     120$
                       09E9  2338
                       09E9  2339 ; Assume that it must be an ENQ
                       09E9  2340
                 0237  30  09E9  2341 30$:    BSBW    DDCMP$SETUP_ENQ_HDR             ; Set up the header for the ENQ
                       09EC  2342                                                  ; This also returns the ENQ
                       09EC  2343                                                  ; block in R8
                   57  95  09EC  2344          TSTB    R7                              ; If NEQ then an hdr error
                   F6  12  09EE  2345          BNEQ    25$
                       09F0  2346
                       09F0  2347 ; Handle an ACK
                       09F0  2348
                   54  01  91  09F0  2349 33$:    CMPB    S^#DDCMP$C_ACK,R4               ; Branch NEQ ACK
                   2D  12  09F3  2350          BNEQ    50$
          50    008C C5  DE  09F5  2351          MOVAL   TF$L_QNAK(R5),R0                ; Get the addr of the NAK block
                       09FA  2352          DSBINT  GF$B_DIPL(R3)                   ; Disable device interrupts
                   00  E5  0A01  2353          BBCC    #XMTQ$V_ONQUEUE,-               ; If BC no NAK on queue
                03 1F  A0  0A03  2354                  XMTQ$B_FLAG(R0),35$
                   50  60  0F  0A06  2355          REMQUE  (R0),R0                         ; If BS on q remove it
                   00  E2  0A09  2356 35$:    BBSS    #XMTQ$V_ONQUEUE,-               ; Is BS then ACK q'd
                04 1F  A2  0A0B  2357                  XMTQ$B_FLAG(R2),40$
              24 B5    62  0E  0A0E  2358          INSQUE  (R2),@TF$Q_CTLQ+4(R5)           ; If BC must q the ACK
                 0C  A5  95  0A12  2359 40$:    TSTB    TF$B_CHAR(R5)                   ; If EQL then FPC no need
                   08  13  0A15  2360          BEQL    48$                             ; to set selection timer
                   0F  E1  0A17  2361 45$:    BBC     #MFD$V_SELECT,-                 ; If BS branch to proc sel exp
                03 01  A1  0A19  2362                  MFD$W_TYPFLG(R1),48$
                    006A  31  0A1C  2363          BRW     100$
                    00A1  31  0A1F  2364 48$:    BRW     120$                            ; Else return
                       0A22  2365
                       0A22  2366 ; Handle NAK message
                       0A22  2367
                54    02  91  0A22  2368 50$:    CMPB    S^#DDCMP$C_NAK,R4               ; Branch in NEQ
                   2C  12  0A25  2369          BNEQ    70$
          50    5C  A5  DE  0A27  2370          MOVAL   TF$L_QACK(R5),R0                ; Get addr of ACK Q block
                       0A2B  2371          DSBINT  GF$B_DIPL(R3)                   ; Disable device interrupts
                   00  E5  0A32  2372          BBCC    #XMTQ$V_ONQUEUE,-               ; If BC then not on Q
                03 1F  A0  0A34  2373                  XMTQ$B_FLAG(R0),55$
                   50  60  0F  0A37  2374          REMQUE  (R0),R0                         ; If BS then rem ACK from Q
                   00  E2  0A3A  2375 55$:    BBSS    #XMTQ$V_UNQUEUE,-               ; If BS msg on queue
                04 1F  A2  0A3C  2376                  XMTQ$B_FLAG(R2),60$
              24 B5    62  0E  0A3F  2377          INSQUE  (R2),@TF$Q_CTLQ+4(R5)           ; Queue the NAK at front of Q
                 0C  A5  95  0A43  2378 60$:    TSTB    TF$B_CHAR(R5)                   ; If EQL then don't start
                   08  13  0A46  2379          BEQL    65$                             ; selection timer
                   0F  E1  0A48  2380          BBC     #MFD$V_SELECT,-                 ; If BC no select to process
                03 01  A1  0A4A  2381                  MFD$W_CNTFLG(R1),65$
```

```
                0039   31   0A4D  2382          BRW     100$                            ; Branch to start timer
                0070   31   0A50  2383 65$:     BRW     120$                            ; Branch to comp queueing
                            0A53  2384
                            0A53  2385 ; Handle REP
                            0A53  2386
                  54   03   91   0A53  2387 70$:     CMPB    S^#DDCMP$C_REP,R4               ; Branch NEQ
                       21   12   0A56  2388          BNEQ    90$
                            0A58  2389          DSBINT  GF$B_DIPL(R3)                   ; Disable device interrupts
                       00   E2   0A5F  2390          BBSS    #XMTQ$V_ONQUEUE,-               ; If BS then entry on queue
                 04 1F A2           0A61  2391                  XMTQ$B_FLAG(R2),75$
                24 B5   62   0E   0A64  2392          INSQUE  (R2),@TF$Q_CTLQ+4(R5)           ; at the top of the queue
                            0A68  2393 75$:
                            0A68  2394
                            0A68  2395 ; The reply timer must be running if a REP is sent, it implies that there is
                            0A68  2396 ; a message on the reply timeout queue.
                            0A68  2397
                16 A5   04   88   0A68  2398          BISB2   #TF$M_TIMER,TF$B_SELTIM(R5)     ; Set timer running
                     0C A5   95   0A6C  2399          TSTB    TF$B_CHAR(R5)                   ; If EQL then don't start
                       05   13   0A6F  2400          BEQL    80$                             ; selection timer
                       0F   E0   0A71  2401          BBS     #MFD$V_SELECT,-                 ; If BS then a select to process
                 13 01 A1           0A73  2402                  MFD$W_CNTFLG(R1),100$
                     004A   31   0A76  2403 80$:     BRW     120$                            ; Branch to comp queueing
                            0A79  2404
                            0A79  2405 ; Assume that message is an STRT or STACK
                            0A79  2406
                            0A79  2407 90$:     DSBINT  GF$B_DIPL(R3)                   ; Disable device interrupts
                       00   E2   0A80  2408          BBSS    #XMTQ$V_ONQUEUE,-               ; If BS then message in on
                 04 1F A2           0A82  2409                  XMTQ$B_FLAG(R2),100$            ; the queue
                24 B5   62   0E   0A85  2410          INSQUE  (R2),@TF$Q_CTLQ+4(R5)           ; Insert message on XMIT Q
                     0C A5   95   0A89  2411 100$:    TSTB    TF$B_CHAR(R5)                   ; Branch EQL the station is a
                       0D   13   0A8C  2412          BEQL    110$                            ; FPC and there is no need to
                            0A8E  2413                                                      ; turn the link
                1F A2   02   A8   0A8E  2414          BISW2   #XMTQ$M_SELECT,XMTQ$B_FLAG(R2)  ; Driver must know to start XMT
                16 A5   02   8A   0A92  2415          BICB2   #TF$M_OWNSELT,TF$B_SELTIM(R5)   ; We have now de-selected
                            0A96  2416                                                      ; ourself.
                       00   E0   0A96  2417          BBS     #TF$V_STATYP,-                  ; Branch BS because the station
                   28 0C A5           0A98  2418                  TF$B_CHAR(R5),120$              ; is a tributary, hence do not
                            0A9B  2419                                                      ; start the selection timer
                            0A9B  2420 110$:    ENBINT                                  ; Enable device interrupts
                     FE26   30   0A9E  2421          BSBW    DDCMP$START_SEL_TIMER
                       08   8A   0AA1  2422          BICB2   #TF$M_RCVDET,-                  ; Reset the RCV detect field
                     16 A5          0AA3  2423                  TF$B_SELTIM(R5)                 ; for a new selection
                       06   EF   0AA5  2424          EXTZV   #DDCMP$C_SEL_THRS,-             ; Extract the field
                       03          0AA7  2425                  #DDCMP$C_THR_SIZE,-             ; to increment
              51 01C5 C5           0AA8  2426                  TF$W_THRES(R5),R1
                   51   07   91   0AAC  2427          CMPB    #DDCMP$C_THR_MAX,R1             ; Has the thres been exceeded
                       07   14   0AAF  2428          BGTR    115$                            ; No
                   57   01   A8   0AB1  2429          BISW2   #DLK$M_TRNSERR,R7               ; Report that thres exceeded
                       51   D4   0AB4  2430          CLRL    R1                              ; Set the SEL threshold count
                            0AB6  2431                                                      ; to zero
                       02   11   0AB6  2432          BRB     118$
                            0AB8  2433
                       51   96   0ABB  2434 115$:    INCB    R1                              ; Incr the thres error counter
                   06  51   F0   0ABA  2435 118$:    INSV    R1,#DDCMP$C_SEL_THRS,-          ; Insert count
                       03          0ABD  2436                  #DDCMP$C_THR_SIZE,-
                   01C5 C5          0ABE  2437                  TF$W_THRES(R5)
                       03   11   0AC1  2438          BRB     123$
```

```
                     0AC3  2439
                     0AC3  2440 120$:   ENBINT                                  ; Enable device interrupts
                  05 0AC6  2441 123$:   RSB
                     0AC7  2442
                     0AC7  2443
13    0A A8    91    0AC7  2444 150$:   CMPB    XMTQ$B_BUFTYP(R8),S^#DYN$C_BUFIO ; If EQL then correct buf
            04    13 0ACB  2445         BEQL    160$
                     0ACD  2446         BUG_CHECK       NOBUFPCKT,FATAL
40 A5       68    0E 0AD1  2447 160$:   INSQUE  (R8),TFSQ_XMTQVF(R5)             ; Insert at top of queue
            05       0AD5  2448         RSB
                     0AD6  2449
```

```
                        0AD6  2451
                        0AD6  2452 ;DDCMP$REQUEUE_XMT
                        0AD6  2453 ;
                        0AD6  2454 ; This routine is called when a message is to be transmitted but T < N + 1
                        0AD6  2455 ; which implies that an ACK was received for some messages but not all that
                        0AD6  2456 ; were sent. This retransmits these messages. These messages already have
                        0AD6  2457 ; headers set up and they can be found on the RTOQ. This routine will
                        0AD6  2458 ; retransmit all messages until T < N + 1 regardless of whether or not
                        0AD6  2459 ; the XMTQ gets filled.
                        0AD6  2460 ;
                        0AD6  2461 ;       INPUTS          R3 = Address of the GF block
                        0AD6  2462 ;                       R5 = Address of the TF block
                        0AD6  2463 ;
                        0AD6  2464 ;       OUTPUTS         R3 and R5 are preserved
                        0AD6  2465 ;                       Entries added to the XMTQ
                        0AD6  2466 ;
                        0AD6  2467 DDCMP$REQUEUE_XMT:
                        0AD6  2468          DSBINT  GF$B_DIPL(R3)                   ; Disable interrupts to DIPL
                        0ADD  2469
                        0ADD  2470 ; This code code is necessary to ensure that messages are not retransmitted
                        0ADD  2471 ; outof order. It is not very elegant because it will fudge around
                        0ADD  2472 ; with the drviers XMT_INPR and POSTQs'. Note that this is all done and DIPL
                        0ADD  2473 ; so that synchronizationis ensured.
                        0ADD  2474
    50    54 A5    D0   0ADD  2475          MOVL    TF$A_XMT_INPR(R5),R0            ; Get the inprogess address
          51   60    D0   0AE1  2476          MOVL    (R0),R1                         ; Get message inprogress
               12    13   0AE4  2477          BEQL    2$                              ; If eql then none
 0D 1F A1      02    E0   0AE6  2478          BBS     #XMTQ$V_CONTROL,XMTQ$B_FLAG(R1),2$ ; If set then control message
               60    D4   0AEB  2479          CLRL    (R0)                            ; Clear message inprogress
               58    DD   0AED  2480          PUSHL   R8
          58   51    D0   0AEF  2481          MOVL    R1,R8
             037A    30   0AF2  2482          BSBW    DDCMP$INS_RTOQ                  ; Put message into RTOQ
               58 8EDO    0AF5  2483          POPL    R8
    50    58 A5    D0   0AF8  2484 2$:       MOVL    TF$A_POSTQ(R5),R0               ; Get address of postq
          51   50    D0   0AFC  2485          MOVL    R0,R1
          51   61    D0   0AFF  2486 4$:      MOVL    (R1),R1                         ; Get next entry
          51   50    D1   0B02  2487 6$:      CMPL    R0,R1
               24    13   0B05  2488          BEQL    8$
 0A A1         13    91   0B07  2489          CMPB    #DYN$C_BUFIO,XMTQ$B_BUFTYP(R1)  ; If NEQ then not an XMT
               F2    12   0B0B  2490          BNEQ    4$
 ED 1F A1      02    E0   0B0D  2491          BBS     #XMTQ$V_CONTROL,XMTQ$B_FLAG(R1),4$ ; If BS then control message
          52   61    D0   0B12  2492          MOVL    (R1),R2                         ; Be sure to save next entry on post
          51   61    0F   0B15  2493          REMQUE  (R1),R1                         ; get the entry
          0103 8F    BB   0B18  2494          PUSHR   #^M<R0,R1,R8>
          58   51    D0   0B1C  2495          MOVL    R1,R8
             034D    30   0B1F  2496          BSBW    DDCMP$INS_RTOQ                  ; Put into its place on RTOQ
          0103 8F    BA   0B22  2497          POPR    #^M<R0,R1,R8>
          51   52    D0   0B26  2498          MOVL    R2,R1                           ; Set next entry to check
               D7    11   0B29  2499          BRB     6$
    52    30 A5    DE   0B2B  2500 8$:       MOVAL   TF$Q_XMTQ(R5),R2                ; Queue the reXMT's at
    51    38 B5    0F   0B2F  2501          REMQUE  @TF$Q_RTOQ(R5),R1               ; Get the entry to reXMT
               6C    1D   0B33  2502          BVS     40$                             ; No more enties on queue
          1B A5    96   0B35  2503          INCB    TF$B_CURRTO(R5)                 ; Increment the number of times
                        0B38  2504                                                   ; a REP has occured
          1B A5    91   0B38  2505          CMPB    TF$B_CURRTO(R5),-               ; Branch on LSS because the rep
          1A A5                0B3B  2506                  TF$B_MAXRTO(R5)             ; T/O max has been reached
               1F    19   0B3D  2507          BLSS    25$                             ; which implies a problem with
```

```
                     0B3F  2508                                                 ; the other node
          13   0A A1  91  0B3F  2509          CMPB    XMTQ$B_BUFTYP(R1),S^#DYN$C_BUFIO ; If EQL then correct buf
                 04  13  0B43  2510          BEQL    10$
                     0B45  2511          BUG_CHECK       NOBUFPCKT,FATAL
          38 A5  61  0E  0B49  2512 10$:     INSQUE  (R1),TF$Q_RTOQ(R5)         ; Ensure buffer completion
               1B A5  94  0B4D  2513          CLRB    TF$B_CURRTO(R5)            ; Reset the cur num of RTO's
             57   02  A8  0B50  2514          BISW2   #DLK$M_PRSTERR,R7
                     0B53  2515          ENBINT
                 50  11  0B56  2516          BRB     50$
          51   38 B5  0F  0B58  2517 20$:     REMQUE  @TF$Q_RTOQ(R5),R1          ; Get the entry to reXMT
                 43  1D  0B5C  2518          BVS     40$                        ; No more enties on queue
          13   0A A1  91  0B5E  2519 25$:     CMPB    XMTQ$B_BUFTYP(R1),S^#DYN$C_BUFIO ; If EQL then correct buf
                 04  13  0B62  2520          BEQL    30$
                     0B64  2521          BUG_CHECK       NOBUFPCKT,FATAL
             50   24 A1  DE  0B68  2522 30$:     MOVAL   XMTQ$B_MSGHDR(R1),R0       ; Get the addr of the msg hdr
       01 A0    8000 8F  AA  0B6C  2523          BICW    #MFD$M_SELECT,MFD$W_CNTFLG(R0) ; Clear select flag
     03 A0   11 A5  90  0B72  2524          MOVB    TF$B_R(R5),MFD$B_RESP(R0)  ; Update the RESP field with
                     0B77  2525                                                 ; the most recent RCV msg
     17 02 A3   00  E1  0B77  2526          BBC     #GF$V_CRC,GF$B_DRVCHR(R3),35$ ; If BC then device does CRC
           0206 8F  BB  0B7C  2527          PUSHR   #^M<R1,R2,R9>
         59   01 A0  B0  0B80  2528          MOVW    MFD$W_CNTFLG(R0),R9        ; Get msg count
       59   C000 8F  AA  0B84  2529          BICW    #MFD$M_SELECT!MFD$M_QSYNC,R9
           52   51  D0  0B89  2530          MOVL    R1,R2                      ; Set buffer addres for CRC calc
                001A  30  0B8C  2531          BSBW    DDCMP$SETCRC               ; Else we must do our own
           0206 8F  BA  0B8F  2532          POPR    #^M<R1,R2,R9>
           62   61  0E  0B93  2533 35$:     INSQUE  (R1),(R2)                  ; Insert onthe XMTQ
           52   51  D0  0B96  2534          MOVL    R1,R2                      ; Set new place to insque after
             10 A5  97  0B99  2535          DECB    TF$B_XQCNT(R5)             ; Grab a slot on the queue
             14 A5  96  0B9C  2536          INCB    TF$B_T(R5)                 ; Increment T (last xmt'd)
                 B7  11  0B9F  2537          BRB     20$
                     0BA1  2538 40$:     ENBINT
    16 A5    04   8A  0BA4  2539 45$:     BICB2   #TF$M_TIMER,TF$B_SELTIM(R5)    ; Set reply timer not running
                 05  0BA8  2540 50$:     RSB
                     0BA9  2541
```

```
                                0BA9  2543
                                0BA9  2544          .SBTTL  Calculate the CRC
                                0BA9  2545
                                0BA9  2546 ;DDCMP$SETCRC - Calulate the CRC for devides that don't support CRC
                                0BA9  2547 ;
                                0BA9  2548 ; This routine calculates CRC's for devices that don't support CRC checking.
                                0BA9  2549 ; Once calculated it stores the CRC in fields on XMTQ it is up to the
                                0BA9  2550 ; device to send the CRC at the proper time.
                                0BA9  2551 ;
                                0BA9  2552 ;                   INPUTS  R2 = Address of XMTQ entry
                                0BA9  2553 ;                           R9 = Length of the data message
                                0BA9  2554 ;
                                0BA9  2555 ;                   OUTPUTS R0 is destroyed
                                0BA9  2556 ;                           R3, R8 and R9 are preserved
                                0BA9  2557 DDCMP$SETCRC:
                      1E   BB   0BA9  2558          PUSHR   #^M<R1,R2,R3,R4>                ; Save registers
                54    52   D0   0BAB  2559          MOVL    R2,R4
        06    00    F4D2 CF   0B   0BAE  2560          CRC     CRCTABLE,#0,#MFD$K_LENGTH,-     ; Calculate CRC for header
                      24   A4        0BB4  2561                  XMTQ$B_MSGHDR(R4)
              20 A4   50   B0   0BB6  2562          MOVW    R0,XMTQ$W_HCRC(R4)              ; Move into HCRC field
              51   24 A4   DE   0BBA  2563          MOVAL   XMTQ$B_MSGHDR(R4),R1            ; Get the address of the data
                 61   05   91   0BBE  2564          CMPB    #DDCMP$C_ENQ,MFD$B_MSGID(R1)    ; If EQL then no need to calc
                      0C   13   0BC1  2565          BEQL    10$                             ; data CRC
  2A A4 59    00    F4BD CF   0B   0BC3  2566          CRC     CRCTABLE,#0,R9,-                ; Calculate the CRC for data
                                0BCB  2567                  XMTQ$K_LENGTH(R4)
              22 A4   50   B0   0BCB  2568          MOVW    R0,XMTQ$W_DCRC(R4)              ; Movw into DCRC field
                      1E   BA   0BCF  2569 10$:     POPR    #^M<R1,R2,R3,R4>                ; Restore registers
                           05   0BD1  2570          RSB
                                0BD2  2571
```

```
                        0BD2  2573
                        0BD2  2574          .SBTTL  Setup headers for maint and data messages
                        0BD2  2575 ; SETUP_MSG_HDR - Set up the header for  a maint or data message
                        0BD2  2576 ;
                        0BD2  2577 ; This routine sets up a header message for a maintenance or data message.
                        0BD2  2578 ; It is passed the buffer which is going to be queued for XMT and inserts
                        0BD2  2579 ; the header into the proper area for transmission.
                        0BD2  2580 ;
                        0BD2  2581 ;       INPUTS          R2 = Address of the buffer for CRC calculation
                        0BD2  2582 ;                       R3 = Address of the GF block
                        0BD2  2583 ;                       R4 = Type of message
                        0BD2  2584 ;                       R5 = Address of the TF block
                        0BD2  2585 ;                       R7 = Clear
                        0BD2  2586 ;                       R8 = Address of the buffer to be queued for transmit
                        0BD2  2587 ;
                        0BD2  2588 ;       OUTPUTS         R1 = Address of header to queue
                        0BD2  2589 ;                       R3, R4, R5, R8, and R9 are preserved.
                        0BD2  2590 ;                       R7 is set if any errors are found
                        0BD2  2591 ;
                        0BD2  2592 DDCMP$SETUP_MSG_HDR:
  51    24 A8    DE     0BD2  2593          MOVAL   XMTQ$B_MSGHDR(R8),R1            ; Get addr of the hdr to setup
     61    54    90     0BD6  2594          MOVB    R4,MFD$B_MSGID(R1)              ; Move in the data msg ID
59    C000 8F    AA     0BD9  2595          BICW2   #<MFD$M_SELECT!-                ; Be sure the count isn't over
                        0BDE  2596                  MFD$M_QSYNC>,R9                 ; a 65K (14 bit field)
  01 A1    59    B0     0BDE  2597          MOVW    R9,MFD$W_CNTFLG(R1)             ; Insert into the count field
  54    81 8F    91     0BE2  2598          CMPB    #DDCMP$C_SOH,R4                 ; Is this a hdr for a data msg
           27    12     0BE6  2599          BNEQ    20$                             ; No
                        0BE8  2600
                        0BE8  2601 ; Yes, set up a data message header
                        0BE8  2602
  06 16 A5    00    E5  0BE8  2603          BBCC    #TF$V_SELECT,TF$B_SELTIM(R5),15$ ; Send select flag??
        8000 8F    A8  0BED  2604          BISW2   #MFD$M_SELECT,-                 ; Set the select flag in header
          01 A1        0BF1  2605                  MFD$W_CNTFLG(R1)
  03 A1    11 A5    90  0BF3  2606 15$:    MOVB    TF$B_R(R5),MFD$B_RESP(R1)       ; Piggyback ACK's in header
     01    12 A5    81  0BF8  2607          ADDB3   TF$B_N(R5),#1,-                 ; Number the data msg
           04 A1        0BFC  2608                  MFD$B_NUMB(R1)
           0E A5    90  0BFE  2609          MOVB    TF$B_XADDR(R5),-                ; Move in trib addr
           05 A1        0C01  2610                  MFD$B_ADDR(R1)                  ; to transmit to
01 A1    4000 8F    A8  0C03  2611          BISW2   #MFD$M_QSYNC,MFD$W_CNTFLG(R1)   ; Set to always resync
  9B 02 A3    00    E0  0C09  2612          BBS     #GF$V_CRC,GF$B_DRVCHR(R3),DDCMP$SETCRC ; IF BS then we do the CRC
              05    0C0E  2613          RSB
                        0C0F  2614
                        0C0F  2615 ; Set up a maintenance message header
                        0C0F  2616
              A8        0C0F  2617 20$:    BISW2   #<MFD$M_QSYNC!-                 ; Set the flags (always ones
                        0C10  2618                  MFD$M_SELECT>,-                 ; when sending main msgs)
01 A1    C000 8F        0C10  2619                  MFD$W_TYPFLG(R1)
                        0C15  2620
                        0C15  2621          ASSUME  DDCMP$C_FILL EQ 0
                        0C15  2622          ASSUME  MFD$B_NUMB EQ MFD$B_RESP+1
                        0C15  2623
          03 A1    B4  0C15  2624          CLRW    MFD$B_RESP(R1)                  ; Zero field
          0E A5    90  0C18  2625          MOVB    TF$B_XADDR(R5),-                ; Set up addr to XMT to
          05 A1        0C1B  2626                  MFD$B_ADDR(R1)
  87 02 A3    00    E0  0C1D  2627          BBS     #GF$V_CRC,GF$B_DRVCHR(R3),DDCMP$SETCRC ; IF BS then we do the CRC
              05    0C22  2628          RSB
                        0C23  2629
```

```
0C23  2630
```

```
                     0C23  2632          .SBTTL  Set up control message headers
                     0C23  2633
                     0C23  2634 ; SETUP_ENQ_HDR - Set up an ENQ message
                     0C23  2635 ;
                     0C23  2636 ; This routine sets up a the appropriate ENQ message. The buffers for each
                     0C23  2637 ; of these messages is stored in the TF block hence no allocation of memory
                     0C23  2638 ; need take place
                     0C23  2639 ;
                     0C23  2640 ;       INPUTS          R3 = Address of the GF block
                     0C23  2641 ;                       R4 = Type of message
                     0C23  2642 ;                       R5 = Address of the TF block
                     0C23  2643 ;                       R7 = Clear
                     0C23  2644 ;
                     0C23  2645 ;       OUTPUTS         R1 = Address of header to queue
                     0C23  2646 ;                       R2 = Address of the block to queue
                     0C23  2647 ;                       R3, R4, and R5 are preserved.
                     0C23  2648 ;                       R7 is set if any errors are found
                     0C23  2649 ;
                     0C23  2650
                     0C23  2651
                     0C23  2652 DDCMP$SETUP_ENQ_HDR:
                     0C23  2653          $DISPATCH       R4,TYPE=B,-
                     0C23  2654          <-      ;index                  ;action
                     0C23  2655          <DDCMP$C_ACK                    20$>-
                     0C23  2656          <DDCMP$C_NAK                    30$>-
                     0C23  2657          <DDCMP$C_REP                    40$>-
                     0C23  2658          <DDCMP$C_STRT                   60$>-
                     0C23  2659          <DDCMP$C_STACK                  70$>-
                     0C23  2660          >
                     0C35  2661
     57    08    A8  0C35  2662 10$:     BISW2   #DLK$M_HDRERR,R7                ; Set up for header error
                 05  0C38  2663          RSB
                     0C39  2664
                     0C39  2665 ; Set up a ACK message
                     0C39  2666
   52    5C A5    DE  0C39  2667 20$:     MOVAL   TF$L_QACK(R5),R2                ; Addr of block to queue
   51    24 A2    DE  0C3D  2668          MOVAL   XMTQ$B_MSGHDR(R2),R1            ; Addr of header
                     0C41  2669          DSBINT  GF$B_DIPL(R3)                   ; Disable interrupts to DIPL
           34    11  0C48  2670          BRB     50$
                     0C4A  2671
                     0C4A  2672
                     0C4A  2673
                     0C4A  2674 ; Set up a NAK message
                     0C4A  2675
 52   008C C5    DE  0C4A  2676 30$:     MOVAL   TF$L_QNAK(R5),R2                ; Addr of block to queue
   51    24 A2    DE  0C4F  2677          MOVAL   XMTQ$B_MSGHDR(R2),R1            ; Addr of header
                     0C53  2678          DSBINT  GF$B_DIPL(R3)                   ; Disable interrupts to DIPL
       0188 C5    F0  0C5A  2679          INSV    TF$B_NAKRSN(R5),-               ; Insert the reason for the NAK
       06   00         0C5E  2680                  #MFD$C_SUBTYP,#MFD$C_SUBTYPL,-
         02 A1         0C60  2681                  MFD$W_TYPFLG+1(R1)
            1A    11  0C62  2682          BRB     50$                             ; Send select flag not set
                     0C64  2683
                     0C64  2684
                     0C64  2685 ; Set up th e REP message
                     0C64  2686
 52   00BC C5    DE  0C64  2687 40$:     MOVAL   TF$L_QREP(R5),R2                ; Addr of block to queue
   51    24 A2    DE  0C69  2688          MOVAL   XMTQ$B_MSGHDR(R2),R1            ; Addr of header
```

```
                                0C6D  2689
                                0C6D  2690          DSBINT  GF$B_DIPL(R3)                    ; Disable interrupts to DIPL
                     03 A1  94  0C74  2691          CLRB    MFD$B_RESP(R1)                  ; Zero field
              04 A1  15 A5  90  0C77  2692          MOVB    TF$B_X(R5),MFD$B_NUMB(R1)       ; Number of last data msg XMT'd
                                0C7C  2693                                                  ; to transmit to
                        08  11  0C7C  2694          BRB     55$
                                0C7E  2695
              03 A1  11 A5  90  0C7E  2696 50$:     MOVB    TF$B_R(R5),MFD$B_RESP(R1)       ; Piggyback ACK's in header
                     04 A1  94  0C83  2697          CLRB    MFD$B_NUMB(R1)                  ; Zero the field
                                0C86  2698
                                0C86  2699
           01 A1  C000 8F  AA  0C86  2700 55$:     BICW    #<MFD$M_SELECT!-                ; Clear the flags in header
                                0C8C  2701                  MFD$M_QSYNC>,MFD$W_CNTFLG(R1)
                        00  E5  0C8C  2702          BBCC    #TF$V_SELECT,-                  ; Send select flag??
                  06 16 A5        0C8E  2703                  TF$B_SELTIM(R5),57$
           01 A1   8000 8F  A8  0C91  2704          BISW2   #MFD$M_SELECT,MFD$W_CNTFLG(R1)  ; Set the select flag in header
           01 A1   4000 8F  A8  0C97  2705 57$:     BISW2   #MFD$M_QSYNC,MFD$W_TYPFLG(R1)   ; Always set QSYNC flag
                     0E A5  90  0C9D  2706          MOVB    TF$B_XADDR(R5),-                ; Move in trib addr
                     05 A1        0CA0  2707                  MFD$B_ADDR(R1)                  ; to transmit to
              03 02 A3   00  E1  0CA2  2708          BBC     #GF$V_CRC,GF$B_DRVCHR(R3),58$   ; If BC then device does CRC
                      FEFF  30  0CA7  2709          BSBW    DDCMP$SETCRC                    ; Else we must do our own
                                0CAA  2710 58$:     ENBINT                                  ; Reset interrupts
                        05  0CAD  2711          RSB
                                0CAE  2712
                                0CAE  2713
                                0CAE  2714
                                0CAE  2715 ; Set up a STRT message
                                0CAE  2716
        52   00EC C5    DE  0CAE  2717 60$:     MOVAL   TF$L_QSTRT(R5),R2               ; Addr of block to queue
          51   24 A2    DE  0CB3  2718          MOVAL   XMTQ$B_MSGHDR(R2),R1            ; Addr of header
                     09  11  0CB7  2719          BRB     80$
                                0CB9  2720
                                0CB9  2721
                                0CB9  2722 ; Set up the STACK message
                                0CB9  2723
        52   011C C5    DE  0CB9  2724 70$:     MOVAL   TF$L_QSTACK(R5),R2              ; Addr of block to queue
          51   24 A2    DE  0CBE  2725          MOVAL   XMTQ$B_MSGHDR(R2),R1            ; Addr of header
                                0CC2  2726
                                0CC2  2727 80$:     DSBINT  GF$B_DIPL(R3)               ; Disable interrupts to DIPL
                     03 A1  B4  0CC9  2728          CLRW    MFD$B_RESP(R1)              ; Zero field
           05 A1     0E A5  90  0CCC  2729          MOVB    TF$B_XADDR(R5),MFD$B_ADDR(R1)   ; Move in trib addr
                                0CD1  2730                                                  ; to transmit to
           03 02 A3   00  E1  0CD1  2731          BBC     #GF$V_CRC,GF$B_DRVCHR(R3),90$   ; If BC then device does CRC
                      FED0  30  0CD6  2732          BSBW    DDCMP$SETCRC                    ; Else we must do our own
                                0CD9  2733 90$:     ENBINT                                  ; Reset interrupts
           16 A5     01  8A  0CDC  2734          BICB2   #TF$M_SELECT,TF$B_SELTIM(R5)    ; Clear set select flag
                        05  0CE0  2735          RSB
                                0CE1  2736
                                0CE1  2737
                                0CE1  2738
```

```
                        OCE1  2740
                        OCE1  2741              .SBTTL  Real clock timer
                        OCE1  2742 ; DDCMP$TIMER - Reply or Selection clock has expired
                        OCE1  2743 ;
                        OCE1  2744 ; This routine handles real time clock timeouts. This routine is called by
                        OCE1  2745 ; the software timer interrupt routine when the clock ticks a second.
                        OCE1  2746 ;
                        OCE1  2747 ;       INPUT           R3 = Address of the UCB
                        OCE1  2748 ;                       R4 = Address of the TFB
                        OCE1  2749 ;                       R5 = Address of the TQE
                        OCE1  2750 ;
                        OCE1  2751 ;       OUTPUTS         FR3 contains the status of the operation
                        OCE1  2752 ;
                        OCE1  2753 ;       SIDE EFFECT:
                        OCE1  2754 ;               A side effect of this routine is on persistant errors
                        OCE1  2755 ;               the line is shutdown and thus the TQE must be removed
                        OCE1  2756 ;               from the timer element queues. The TQE must be removed for
                        OCE1  2757 ;               two reasons: 1) There may be a time when the driver accessing
                        OCE1  2758 ;               the protocol must deallocate the TF/GF blocks on line shut
                        OCE1  2759 ;               down. Since the TQE is part of those blocks then we can no
                        OCE1  2760 ;               longer use that TQE for our timer. 2) We would like the
                        OCE1  2761 ;               protocol timer to be running only when the protocol is
                        OCE1  2762 ;               running to eliminate some overhead on the system. This will
                        OCE1  2763 ;               not happen obviously becuase at this point the TQE is not on the
                        OCE1  2764 ;               timer queue. Therefore a BIT was established to determine
                        OCE1  2765 ;               if indeed the timer was stopped. If we find that
                        OCE1  2766 ;               the timer is not longer running then we should not
                        OCE1  2767 ;               return with our TQE, but instead return with the
                        OCE1  2768 ;               system NOREPT TQE.  This TQE was established just for that
                        OCE1  2769 ;               purpose a description of its use is included in
                        OCE1  2770 ;               the module EXE$SWTIMINT.
                        OCE1  2771 ;
                        OCE1  2772 ;       NOTE:
                        OCE1  2773 ;               This routine expects the R0 to have the LBS if the call
                        OCE1  2774 ;               to the driver timers' routine did not cause the device
                        OCE1  2775 ;               to be shut down. If the device was shut down then the
                        OCE1  2776 ;               TF/GF block could potentially be deallocated and thus
                        OCE1  2777 ;               DDCMP had better know not to try to access that block.
                        OCE1  2778 DDCMP$TIMER::
      03FC 8F    BB     OCE1  2779              PUSHR   #^M<R2,R3,R4,R5,R6,R7,R8,R9>
      52   55    D0     OCE5  2780              MOVL    R5,R2                           ; Put addr of TQE into R2
51    10   A2    D0     OCE8  2781              MOVL    TQE$L_FR3(R2),R1                ; Get the UCB address
55    14   A2    D0     OCEC  2782              MOVL    TQE$L_FR4(R2),R5                ; Get the TF block address
53    4C   A5    D0     OCF0  2783              MOVL    TF$A_GFB(R5),R3                 ; Get GF block address
      11   A3    D4     OCF4  2784              CLRL    GF$L_TQE_STS(R3)                ; Use for status
                        OCF7  2785
                        OCF7  2786              ASSUME  DDCMP$C_STA_HLT EQ 0
                        OCF7  2787
            63  95      OCF7  2788              TSTB    GF$B_STATE(R3)                  ; If protocol halted no work
            37  13      OCF9  2789              BEQL    10$
12 0C A5    00  E0      OCFB  2790              BBS     #TF$V_STATYP,TF$B_CHAR(R5),5$   ; If Trib don't use for startup
                        0D00  2791
      63    02  91      0D00  2792              CMPB    #DDCMP$C_STA_IST,GF$B_STATE(R3) ; Branch EQL because selection
            40  13      0D03  2793              BEQL    DDCMP$SELECT                    ; timer is running
      63    03  91      0D05  2794              CMPB    #DDCMP$C_STA_AST,GF$B_STATE(R3) ; Branch EQL because selection
            38  13      0D08  2795              BEQL    DDCMP$SELECT                    ; timer is running
        0C  A5  95      0D0A  2796              TSTB    TF$B_CHAR(R5)                   ; Is this a FPC line?
```

```
                26   12   0D0D  2797          BNEQ    20$                                     ; Branch if anything else
              00DB   31   0D0F  2798          BRW     DDCMP$REP_TO                            ; Branch to handle as a reply
                                  0D12  2799                                                  ; timer
            05 A3   B7   0D12  2800 5$:      DECW    GF$W_NEXTCNT(R3)                        ; Decr by one for acurate cnt
                0E   14   0D15  2801          BGTR    8$
      09 16 A5   01   E1   0D17  2802          BBC     #TF$V_OWNSELT,TF$B_SELTIM(R5),8$ ; Branch if we do not own line
       0189 C5   02   88   0D1C  2803          BISB    #TF$M_SACK,TF$B_SFLAGS(R5)              ; Set to send the ACK
         16 A5   01   88   0D21  2804          BISB    #TF$M_SELECT,TF$B_SELTIM(R5)            ; with the selection timer set
            50   55   D0   0D25  2805 8$:      MOVL    R5,R0                                   ; Save the TF block address
            55   51   D0   0D28  2806          MOVL    R1,R5                                   ; Get UCB address in R5
      51    11 A3   D0   0D2B  2807          MOVL    GF$L_TQE_STS(R3),R1                     ; Set TQE status to driver
            50 B0   16   0D2F  2808          JSB     @TF$A_DEV_TIMER(R0)                     ; Inform driver of more work
              0122   31   0D32  2809 10$:     BRW     DDCMP$TIMER_END
                                  0D35  2810
         16 A5   01   E1   0D35  2811 20$:     BBC     #TF$V_OWNSELT,TF$B_SELTIM(R5),- ; Branch if we do not own line
                 0B        0D39  2812                  DDCMP$SELECT
         16 A5   01   88   0D3A  2813          BISB    #TF$M_SELECT,TF$B_SELTIM(R5)            ; with the selection timer set
       0189 C5   02   88   0D3E  2814          BISB    #TF$M_SACK,TF$B_SFLAGS(R5)              ; Set to send the ACK
                 E0   11   0D43  2815          BRB     8$
                                  0D45  2816
                                  0D45  2817 DDCMP$SELECT:
   50    00000000'GF   D0   0D45  2818          MOVL    G^EXE$GL_ABSTIM,R0                      ; Get time
         0D A3   50   D1   0D4C  2819          CMPL    R0,GF$L_SELEND(R3)                      ; If NEQ then timer hasn't
                 63   15   0D50  2820          BLEQ    18$                                     ; expired - Compare seconds
    11 A3   8000 8F   A8   0D52  2821          BISW2   #DLK$M_TMREXPD,GF$L_TQE_STS(R3) ; Set the select time has expd
                 51   DD   0D58  2822          PUSHL   R1
      51   018E C5   3E   0D5A  2823          MOVAW   TF$B_STO(R5),R1                         ; Incr the seltimeout field
              0374   30   0D5F  2824          BSBW    DDCMP$INCR_COUNTER
                 51 8ED0   0D62  2825          POPL    R1
                 03   E1   0D65  2826          BBC     #TF$V_RCVDET,-                  ; Branch BS RCV was detected
        07 16 A5        0D67  2827                  TF$B_SELTIM(R5),5$
                 01   A8   0D6A  2828          BISW2   #TF$M_NOREP_SEL,-               ; Else set bit no reply to sel
         01BC C5        0D6C  2829                  TF$W_STOBC(R5)
                 05   11   0D6F  2830          BRB     7$
                 02   A8   0D71  2831 5$:      BISW2   #TF$M_INCREP_SEL,-              ; Set attempt to reply was
         01BC C5        0D73  2832                  TF$W_STOBC(R5)                          ; detected
                 06   EF   0D76  2833 7$:      EXTZV   #DDCMP$C_SEL_THRS,-             ; Extract the field
                 03        0D78  2834                  #DDCMP$C_THR_SIZE,-             ; to increment
   54    01C5 C5        0D79  2835                  TF$W_THRES(R5),R4
                 54   96   0D7D  2836          INCB    R4                                      ; Incr the thres error counter
         54      07   91   0D7F  2837          CMPB    #DDCMP$C_THR_MAX,R4                     ; Has the thres been exceeded
                 06   14   0D82  2838          BGTR    10$                                     ; No
       11 A3     01   A8   0D84  2839          BISW2   #DLK$M_TRNSERR,GF$L_TQE_STS(R3) ; Report that thres exceeded
                 54   D4   0D88  2840          CLRL    R4                                      ; Clear the field
         06      54   F0   0D8A  2841 10$:     INSV    R4,#DDCMP$C_SEL_THRS,-          ; Insert the count
                 03        0D8D  2842                  #DDCMP$C_THR_SIZE,-
         01C5 C5        0D8E  2843                  TF$W_THRES(R5)
                                  0D91  2844
                                  0D91  2845 ; Incr the number of times that a seletion interval has timed out on this
                                  0D91  2846 ; station
                                  0D91  2847
            08 A3   96   0D91  2848          INCB    GF$B_CURSEL(R3)
         54    63   9A   0D94  2849          MOVZBL  GF$B_STATE(R3),R4                       ; Get protocol state
                                  0D97  2850
                                  0D97  2851 ; Branch to handle selection timeout according to the state the protocol
                                  0D97  2852 ; is currently in.
                                  0D97  2853
```

```
                        0D97  2854          DISPATCH R4,TYPE=B,-
                        0D97  2855          <-      ; Function              ; Action
                        0D97  2856                  <DDCMP$C_STA_RUN        25$>,-
                        0D97  2857                  <DDCMP$C_STA_IST        15$>,-
                        0D97  2858                  <DDCMP$C_STA_AST        20$>,-
                        0D97  2859                  <DDCMP$C_STA_MNT        25$>,-
                        0D97  2860          >
                        0DA3  2861          BUG_CHECK       NOBUFPCKT,FATAL
                        0DA7  2862
                        0DA7  2863 ; If this is in the IST state, which means that the device hasn't received
                        0DA7  2864 ; some sort of acknowledgement of life from the other end then use
                        0DA7  2865 ; the STRTSEL. This field was put in to allow for more time on the initial
                        0DA7  2866 ; startup for things like dialing a call over real modems. If the
                        0DA7  2867 ; SELECTION TIMER for the device is set to 3 seconds then using this
                        0DA7  2868 ; field will give the user about 60 seconds to dial the call before
                        0DA7  2869 ; tne line is dropped.
                        0DA7  2870
      54   06   9A      0DA7  2871 15$:    MOVZBL  #DDCMP$C_STRT,R4                ; Set that a STRT must be sent
2B A3    08   A3   91    0DAA  2872         CMPB    GF$B_CURSEL(R3),GF$B_STRTSEL(R3); Branch on LSS because the max
           1F   19      0DAF  2873         BLSS    30$                             ; SEL T/O has been not reached
   11 A3   02   A8      0DB1  2874         BISW2   #DLK$M_PRSTERR,GF$L_TQE_STS(R3) ; Report to driver a peristant
           27   11      0DB5  2875 18$:    BRB     35$
                        0DB7  2876
      54   07   9A      0DB7  2877 20$:    MOVZBL  #DDCMP$C_STACK,R4               ; Set that a STACK must be sent
           07   11      0DBA  2878         BRB     27$
                        0DBC  2879
   16 A5   01   88      0DBC  2880 25$:    BISB    #TF$M_SELECT,TF$B_SELTIM(R5)    ; Set to send a select flag
      54   01   9A      0DC0  2881         MOVZBL  #DDCMP$C_ACK,R4                 ; Force sending an ACK
07 A3    08   A3   91    0DC3  2882 27$:    CMPB    GF$B_CURSEL(R3),GF$B_MAXSEL(R3) ; Branch on LSS because the max
           06   19      0DC8  2883         BLSS    30$                             ; SEL T/O has been not reached
   11 A3   02   A8      0DCA  2884         BISW2   #DLK$M_PRSTERR,GF$L_TQE_STS(R3) ; Report to driver a peristant
           0E   11      0DCE  2885         BRB     35$
                        0DD0  2886
           57   D4      0DD0  2887 30$:    CLRL    R7                              ; Clear error reg
           51   DD      0DD2  2888         PUSHL   R1                              ; Save R1
         FBB2   30      0DD4  2889         BSBW    DDCMP$QUEUE_XMT                 ; Queue the transmit
           51 8ED0      0DD7  2890         POPL    R1                              ; Restore R1
   11 A3   57   A8      0DDA  2891         BISW    R7,GF$L_TQE_STS(R3)             ; Set any errors on queueing
      50   55   D0      0DDE  2892 35$:    MOVL    R5,R0                           ; Save TF block address
      55   51   D0      0DE1  2893         MOVL    R1,R5                           ; Get UCB address in R5
   51   11 A3   D0      0DE4  2894         MOVL    GF$L_TQE_STS(R3),R1             ; Set TQE status to driver
        50 B0   16      0DE8  2895         JSB     @TF$A_DEV_TIMER(R0)             ; Inform driver of more work
           6A   11      0DEB  2896         BRB     DDCMP$TIMER_END
                        0DED  2897
                        0DED  2898 DDCMP$REP_TO:
50 00000000'GF   D0     0DED  2899         MOVL    G^EXE$GL_ABSTIM,R0              ; Get time
      54   38 A5   DE    0DF4  2900         MOVAL   TF$Q_RTOQ(R5),R4                ; Get the addr of the queue
      58   38 A5   D0    0DF8  2901         MOVL    TF$Q_RTOQ(R5),R8                ; Get the first entry on queue
         58   54   D1    0DFC  2902         CMPL    R4,R8                           ; Branch on EQL queue is empty
              49   13    0DFF  2903         BEQL    20$
   10 A8   50   D1      0E01  2904         CMPL    R0,XMTQ$L_TIMEND(R8)            ; Branch NEQ timer hasn't exprd
              43   15    0E05  2905         BLEQ    20$
              12   BB    0E07  2906         PUSHR   #^M<R1,R4>                      ; Save R1
      54   03   9A      0E09  2907         MOVZBL  #DDCMP$C_REP,R4                 ; Set up to queue the REP
           57   D4      0E0C  2908         CLRL    R7                              ; Clear error reg
         FB78   30      0E0E  2909         BSBW    DDCMP$QUEUE_XMT                 ; Queue the transmit
              12   BA    0E11  2910         POPR    #^M<R1,R4>                      ; Restore R1
```

```
                11 A3    57    A8  0E13  2911          BISW    R7,GF$L_TQE_STS(R3)             ; Set any errors on queueing
                50   18 A5    3C  0E17  2912          MOVZWL  TF$W_REPWAIT(R5),R0             ; Get reply wait time
                   10 A8    C1  0E1B  2913 10$:      ADDL3   XMTQ$L_TIMEND(R8),-             ; Get new expiration time
                10 A8    50      0E1E  2914                  R0,XMTQ$L_TIMEND(R8)
                   58    68  D0  0E21  2915          MOVL    (R8),R8                         ; Get next entry
                   58    54  D1  0E24  2916          CMPL    R4,R8                           ; Branch EQL end of queue
                         F2  12  0E27  2917          BNEQ    10$
                         51  DD  0E29  2918          PUSHL   R1                              ; Save the register
             51    01C1 C5  3E  0E2B  2919          MOVAW   TF$B_LRTO(R5),R1                ; Incr the num of local reply
                      02A3  30  0E30  2920          BSBW    DDCMP$INCR_COUNTER              ; timeouts
                      036C  30  0E33  2921          BSBW    DDCMP$INCR_XMT_THRS             ; Incr the XMT thres counter
                         51 8ED0  0E36  2922          POPL    R1                              ; Restore the register
                   1B A5    96  0E39  2923          INCB    TF$B_CURRTO(R5)                 ; Incr the number of times that
                                0E3C  2924                                                  ; a RTO has been snt on the msg
                   1B A5    91  0E3C  2925          CMPB    TF$B_CURRTO(R5),-               ; Branch on LSS because the max
                   1A A5        0E3F  2926                  TF$B_MAXRTO(R5)                 ; RTO has not been reached
                         07  19  0E41  2927          BLSS    20$
                   1B A5    94  0E43  2928          CLRB    TF$B_CURRTO(R5)                 ; Reset the cur num of RTO's
                11 A3    02  A8  0E46  2929          BISW2   #DLK$M_PRSTERR,GF$L_TQE_STS(R3) ; Report to driver a peristant
                50   55  D0  0E4A  2930 20$:      MOVL    R5,R0                           ; Save TF block address
                55   51  D0  0E4D  2931          MOVL    R1,R5                           ; Get UCB address
                51   11 A3  D0  0E50  2932          MOVL    GF$L_TQE_STS(R3),R1             ; Set TQE status to driver
                   50 B0    16  0E54  2933          JSB     @TF$A_DEV_TIMER(R0)             ; Poke the device with work
                                0E57  2934
                                0E57  2935 DDCMP$TIMER_END:
                      03FC 8F   BA  0E57  2936          POPR    #^M<R2,R3,R4,R5,R6,R7,R8,R9>    ; Restore registers
                         09 50  E9  0E5B  2937          BLBC    R0,5$                           ; If LBC then device shut down
                51   4C A4  D0  0E5E  2938          MOVL    TF$A_GFB(R4),R1                 ; Get GFB address
                         00  E0  0E62  2939          BBS     #GF$V_TIMER_RUNNING,-           ; If BS then timer running
                   07 01 A1      0E64  2940                    GF$B_TIMER_STATE(R1),10$      ; Else make sure the old
             55    00000000'GF  DE  0E67  2941 5$:       MOVAL   G^EXE$AL_TQENOREPT,R5           ; TQE is not returned to the
                                05  0E6E  2942 10$:      RSB                             ; caller
                                0E6F  2943
                                0E6F  2944
                                0E6F  2945
```

```
                                2947
                                2948          .SBTTL  Reply timeout clock queues and timer handling
                                2949
                                2950 ; TIMER - Reply and selection timeout clock
                                2951 ;
                                2952 ; This routine is responsible for the real time clock associated with this
                                2953 ; protocol. The TQE is part of the TF block hence no need to worry about
                                2954 ; allocating and deallocating it. The TQE is set up only once at controller
                                2955 ; init time. It ticks once per .5 seconds while the system is running. In this
                                2956 ; way the driver can use the system clock to time selection intervals and
                                2957 ; reply timeouts, but not use large pieces of system non-paged pool. Half
                                2958 ; duplex and multipoint stations use the TQE as aselection timer. Full
                                2959 ; duplex uses the TQE for a reply timeout clock.
                                2960 ;
                                2961 ; This is also reponsible for queueing messages for reply timeout. Handling
                                2962 ; these timeouts is different for each type of station.
                                2963 ;
                                2964 ;       FPC     =       Real time clock
                                2965 ;       HPC     =       End of next selection interval
                                2966 ;       FMC     =       Before next selection interval
                                2967 ;       HMC     =       Before next selection interval
                                2968 ;       FMT     =       Before next selection interval
                                2969 ;       HMT     =       Before next selection interval
                                2970 ;
                                2971 ;
                                2972 ;       INPUTS          R3 = Address of the GF block
                                2973 ;                       R5 = TF block
                                2974 ;                       R8 = Address of messge to reply timeout
                                2975 ;
                                2976 ;       OUTPUTS         R3,R5 and R8 are preserved
                                2977 ;       Addition or deletion form the RTOQ
                                2978 ;
                                2979 DDCMP$INS_RTOQ:
                     1F   BB    2980          PUSHR   #^M<R0,R1,R2,R3,R4>             ; Save registers
 50   00000000'GF     D0    2981          MOVL    G^EXE$GL_ABSTIM,R0              ; Get the addr of timer buffer
       51   18 A5    3C    2982          MOVZWL  TF$W_REPQAI(R5),R1              ; Get reply wait time
  10 A8   51   50    C1    2983          ADDL3   R0,R1,XMTQ$L_TIMEND(R8)         ; Set time to expire
                     OE81   2984
       52   24 A8    9E    2985          MOVAB   XMTQ$B_MSGHDR(R8),R2            ; get message header
       53   38 A5    9E    2986          MOVAB   TF$Q_RTOQ(R5),R3                ; Get start of reply timeout q
          50   53    D0    2987          MOVL    R3,R0                           ; Save address on R0
          60   50    D1    2988          CMPL    R0,(R0)                         ; If EQL then empy queue
                17    13    2989          BEQL    10$
       50   04 A0    D0    2990 5$:      MOVL    4(R0),R0                        ; Get previous entry
          53   50    D1    2991          CMPL    R0,R3                           ; If EQL then end of queue
                0E    13    2992          BEQL    10$
       51   24 A0    9E    2993          MOVAB   XMTQ$B_MSGHDR(R0),R1            ; Get entries message header
54   04 A2   04 A1    83    2994          SUBB3   MFD$B_NUMB(R1),MFD$B_NUMB(R2),R4 ; If LSS then do not queue after
                EB    19    2995          BLSS    5$
                0D    13    2996          BEQL    15$
          60   68    0E    2997 10$:     INSQUE  (R8),(R0)                       ; Put on RTOQ
       16 A5   04    88    2998          BISB2   #TF$M_TIMER,TF$B_SELTIM(R5)      ; Set timer running
       13   0A A8    91    2999          CMPB    XMTQ$B_BUFTYP(R8),S^#DYN$C_BUFIO ; If EQL then correct buf
                04    13    3000          BEQL    20$
                      0E    3001 15$:     BUG_CHECK       NOBUFPCKT,FATAL
                1F    BA    3002 20$:     POPR    #^M<R0,R1,R2,R3,R4>             ; Restore registers
                      05    3003          RSB
```

```
                        OEBC  3004
                        OEBC  3005 ;DDCMP$REM_RTOQ - Remove ACk'd XMTs from timer queue
                        OEBC  3006 ;
                        OEBC  3007 ; This routine is called to stop reply timing on ACK'd messages. First all
                        OEBC  3008 ; ACK'd entries are removed from the queue. If then queue becomes empty the
                        OEBC  3009 ; bit indicating the timer is running is cleared, otherwise the bit is
                        OEBC  3010 ; left on.
                        OEBC  3011 ;
                        OEBC  3012 ;       INPUTS          R3 = Address of the GF block
                        OEBC  3013 ;                       R5 = Address of the TF block
                        OEBC  3014 ;                       R8 = Address of the receive message
                        OEBC  3015 ;
                        OEBC  3016 ;       OUTPUTS         R1,R2,R3,R5 and R8 are preserved
                        OEBC  3017 ;
                        OEBC  3018 DDCMP$REM_RTOQ:
                  06    BB    OEBC  3019          PUSHR   #^M<R1,R2>
          51    38 B5   0F    OEBE  3020 2$:      REMQUE  @TF$Q_RTOQ(R5),R1               ; Get the next entry on RTOQ
                40 1D   91    OEC2  3021          BVS     10$                             ; If VS then queue empty
          13    0A A1   04    OEC4  3022          CMPB    XMTQ$B_BUFTYP(R1),S^#DYN$C_BUFIO ; If EQL then correct buf
                        13    OEC8  3023          BEQL    5$
                        OECA  3024          BUG_CHECK       NOBUFPCKT,FATAL
          52    24 A1   DE    OECE  3025 5$:      MOVAL   XMTQ$B_MSGHDR(R1),R2            ; The addr of the msg hdr
                        OED2  3026
                        OED2  3027 ; Check the condition NUMB <= A
                        OED2  3028
    50  04 A2   13 A5   83    OED2  3029          SUBB3   TF$B_A(R5),MFD$B_NUMB(R2),R0    ; Branch if A < NUMB
                31 14   OED8  3030          BGTR    20$                             ; No more to take off queue
                18 A5   94    OEDA  3031          CLRB    TF$B_CURRTO(R5)                 ; Clear the count of rep's
          57  0400 8F   A8    OEDD  3032          BISW    #DLK$M_XMTCMP,R7                ; Inform the driver there is
                        OEE2  3033                                                      ; an I/O to complete
            13  0A A1  91    OEE2  3034          CMPB    XMTQ$B_BUFTYP(R1),S^#DYN$C_BUFIO ; If EQL then correct buf
                04 13   OEE6  3035          BEQL    8$
                        OEE8  3036          BUG_CHECK       NOBUFPCKT,FATAL
        2C B5   61 0E   OEEC  3037 8$:      INSQUE  (R1),@TF$Q_CMPQ+4(R5)           ; Insert request on CMPQ
        50  01 A2  3C    OEF0  3038          MOVZWL  MFD$W_CNTFLG(R2),R0             ; Get the count
    50  C000 8F   AA    OEF4  3039          BICW    #<MFD$M_SELECT!-                ; Clear if flags are set
                        OEF9  3040                  MFD$M_QSYNC>,R0
            019E C5   D6  OEF9  3041          INCL    TF$L_DMSGX(R5)                  ; Incr the number of msgs XMT'd
    0192 C5     50 C0   OEFD  3042          ADDL2   R0,TF$L_DBYTX(R5)               ; Incr the number of byte XMT'd
                BA 11   OF02  3043          BRB     2$                              ; Get next entry
                04 8A   OF04  3044 10$:     BICB2   #TF$M_TIMER,-                  ; Set timer no longr running
                  16 A5         OF06  3045          TF$B_SELTIM(R5)
                06 BA   OF08  3046          POPR    #^M<R1,R2>
                05 OF0A 3047          RSB
                        OF0B  3048
        13    0A A1   91    OF0B  3049 20$:     CMPB    XMTQ$B_BUFTYP(R1),S^#DYN$C_BUFIO ; If EQL then correct buf
                04 13   OF0F  3050          BEQL    30$
                        OF11  3051          BUG_CHECK       NOBUFPCKT,FATAL
        38 A5   61 0E   OF15  3052 30$:     INSQUE  (R1),TF$Q_RTOQ(R5)              ; Insert the entry back on RTOQ
                06 BA   OF19  3053          POPR    #^M<R1,R2>
                05 OF1B 3054          RSB
                        OF1C  3055
```

```
                        0F1C  3057          .SBTTL  Halt
                        0F1C  3058
                        0F1C  3059 ;DDCMP$HALT
                        0F1C  3060 ;
                        0F1C  3061 ; This routine halts the protocol. The only action halting the protocol
                        0F1C  3062 ; is to stop any real time clocks which may be running and returns all
                        0F1C  3063 ; outstanding transmits.
                        0F1C  3064 ;
                        0F1C  3065 ;       INPUTS          R3 = Address of the GF block
                        0F1C  3066 ;                       R5 = Address of the TF block
                        0F1C  3067 ;                       R8 = Address of the data message header if given
                        0F1C  3068 ;
                        0F1C  3069 ;       OUTPUTS         R3 and R5 are preserved
                        0F1C  3070 ;                       R6 and R7 contian return status to driver
                        0F1C  3071 ;
                        0F1C  3072 DDCMP$HALT:
        54   08   3C   0F1C  3073          MOVZWL  S^#DLK$C_RETINFO,R6
             57   D4   0F1F  3074          CLRL    R7
      0189 C5   94   0F21  3075          CLRB    TF$B_SFLAGS(R5)                 ; Clear flags
             58   D5   0F25  3076          TSTL    R8                              ; Branch NEQ because this halt
                  1A   13   0F27  3077          BEQL    10$                             ; was not initiated by RCV'ing
                       0F29  3078                                                  ; a STRT
           F8F5   30   0F29  3079          BSBW    DDCMP$CHK_HDR                   ; First make sure the message
                       0F2C  3080                                                  ; was ok
             57   D5   0F2C  3081          TSTL    R7                              ; If EQL then message CRC ok
                  03   13   0F2E  3082          BEQL    5$
            F2FB   31   0F30  3083          BRW     DDCMP$ERROR                     ; Else ignore message
57  0040 8F   3C   0F33  3084 5$:      MOVZWL  #DLK$M_STRTRCV,R7               ; Assume that a STRT was rcvd
  68   90 8F   91   0F38  3085          CMPB    #DDCMP$C_DLE,MFD$B_MSGID(R8)    ; If NEQ then a STRT was rcv'd
                  05   12   0F3C  3086          BNEQ    10$
57  0080 8F   3C   0F3E  3087          MOVZWL  #DLK$M_MNTRCV,R7                ; Else a DLE was received
                       0F43  3088 10$:
                       0F43  3089 DDCMP$MAINT_HALT:                            ; Complete XMT's when MOP mode
          16 A5   04   8A   0F43  3090          BICB2   #TF$M_TIMER,TF$B_SELTIM(R5)     ; Clear bit to set timer off
        52   014C C5   DE   0F47  3091          MOVAL   TF$L_TQE(R5),R2                 ; Get TQE address
20 A2 00000000 01C9C380 8F   7D   0F4C  3092          MOVQ    #DDCMP$C_STPTQE,TQE$Q_DELTA(R2) ; Set delta for halted protocol
          51   38 B5   0F   0F58  3093 20$:      REMQUE  @TF$Q_RTOQ(R5),R1               ; Send all outstanding XMT's
                       0F5C  3094                                                  ; to I/O complete
                  10   1D   0F5C  3095          BVS     25$                             ; If VS then Q empty
          13   0A A1   91   0F5E  3096          CMPB    XMTQ$B_BUFTYP(R1),S^#DYN$C_BUFIO ; If EQL then correct buf
                  04   13   0F62  3097          BEQL    22$
                       0F64  3098          BUG_CHECK       NOBUFPCKT,FATAL
          2C B5   61   0E   0F68  3099 22$:      INSQUE  (R1),@TF$Q_CMPQ+4(R5)
                  EA   11   0F6C  3100          BRB     20$
                       0F6E  3101 25$:      DSBINT  GF$B_DIPL(R3)                   ; Lock out device interrupts
          51   30 B5   0F   0F75  3102 30$:      REMQUE  @TF$Q_XMTQ(R5),R1               ; Send all outstanding XMT's
                       0F79  3103                                                  ; to I/O complete
                  18   1D   0F79  3104          BVS     40$                             ; If VS then Q empty
          13   0A A1   91   0F7B  3105          CMPB    XMTQ$B_BUFTYP(R1),S^#DYN$C_BUFIO ; If EQL then correct buf
                  04   13   0F7F  3106          BEQL    35$
                       0F81  3107          BUG_CHECK       NOBUFPCKT,FATAL
                  02   E0   0F85  3108 35$:      BBS     #XMTQ$V_CONTROL,-               ; If BS then control msg
             EB 1F A1        0F87  3109                  XMTQ$B_FLAG(R1),30$             ; don't sendto I/O complete
          2C B5   61   0E   0F8A  3110          INSQUE  (R1),@TF$Q_CMPQ+4(R5)
                10 A5   96   0F8E  3111          INCB    TF$B_XQCNT(R5)                  ; Free the next spot on queue
                  E2   11   0F91  3112          BRB     30$
                       0F93  3113 40$:      ENBINT                                  ; Enable interrupts
```

```
                51    40 B5    0F   0F96  3114 45$:    REMQUE  @TF$Q_XMTOVF(R5),R1             ; Send all XMTs on overflow q
                         10    1D   0F9A  3115          BVS     50$                            ; to I/O complete
                13    0A A1    91   0F9C  3116          CMPB    XMTQ$B_BUFTYP(R1),S^#DYN$C_BUFIO ; If EQL then correct buf
                         04    13   0FA0  3117          BEQL    43$
                                    0FA2  3118          BUG_CHECK       NOBUFPCKT,FATAL
                2C B5    61    0E   0FA6  3119 43$:     INSQUE  (R1),@TF$Q_CMPQ+4(R5)
                         EA    11   0FAA  3120          BRB     45$
                               05   0FAC  3121 50$:     RSB
                                    0FAD  3122
```

```
                        OFAD  3124          .SBTTL  Maintenance mode
                        OFAD  3125 ; DDCMP$MAINT - Maintenance mode
                        OFAD  3126 ;
                        OFAD  3127 ; This routine is called when the protocol is in maintenance mode or when a
                        OFAD  3128 ; maintenance msg has been received ot must be sent.  Maintenance mode
                        OFAD  3129 ; performs only the framming and link management portions of DDCMP.
                        OFAD  3130 ;
                        OFAD  3131 ;       INPUTS          R3 = Address of the GF block
                        OFAD  3132 ;                       R5 = Address of the TF block
                        OFAD  3133 ;                       R6 = Contain the action to be executed
                        OFAD  3134 ;                       R7 = Any bits set then message was in error
                        OFAD  3135 ;                       R8 = Address of maintenance message header
                        OFAD  3136 ;                       R9 = Contians the number of bytes to send
                        OFAD  3137 ;
                        OFAD  3138 ;       OUTPUTS         R6,R7 = Contain status of the action
                        OFAD  3139 ;                       R3, R5 and R8 are preserved
                        OFAD  3140 ;                       R9 is the size of the maintenance message received
                        OFAD  3141 ;
                        OFAD  3142 ;       The XMTQ may have an entry on it.
                        OFAD  3143 ;
                        OFAD  3144 DDCMP$MAINT:
          56    02  91  OFAD  3145          CMPB    S^#DLK$C_XMTMSG,R6              ; Branch if EQL xmtmsg
                1C  13  OFB0  3146          BEQL    20$
          56    08  3C  OFB2  3147          MOVZWL  S^#DLK$C_RETINFO,R6             ; Set up return to driver
                57  D4  OFB5  3148          CLRL    R7                              ; Clear all set bits
             F867  30  OFB7  3149          BSBW    DDCMP$CHK_HDR                   ; Check the header
                57  D5  OFBA  3150          TSTL    R7                              ; Branch NEQ errors in header
                2D  12  OFBC  3151          BNEQ    30$
    57    4100 8F  A8  OFBE  3152          BISW2   #<DLK$M_RCVACK!-                ; Set rcv msg is ok
                       OFC3  3153                  DLK$M_TRNLK>,R7                 ; Set to turn link
      59      01 A8  B0  OFC3  3154          MOVW    MFD$W_CNTFLG(R8),R9             ; Get the size of the
    59    C000 8F  AA  OFC7  3155          BICW    #<MFD$M_SELECT!-                ; of the message received
                       OFCC  3156                  MFD$M_QSYNC>,R9
                1D  11  OFCC  3157          BRB     30$
          56    08  3C  OFCE  3158 20$:     MOVZWL  #DLK$C_RETINFO,R6               ; Set up return to driver
                57  D4  OFD1  3159          CLRL    R7                              ; Clear all set bits
          63    01  91  OFD3  3160          CMPB    #DDCMP$C_STA_RUN,GF$B_STATE(R3) ; If NEQ then no XMT's are
                03  12  OFD6  3161          BNEQ    25$                             ; outstanding
             FF68  30  OFD8  3162          BSBW    DDCMP$MAINT_HALT                ; Branch to send XMT's to comp
       54    90 8F  9A  OFDB  3163 25$:     MOVZBL  #DDCMP$C_DLE,R4                 ; Set to queue a maint msg
             F9A7  30  OFDF  3164          BSBW    DDCMP$QUEUE_XMT                 ; Queue the msg
                57  95  OFE2  3165          TSTB    R7                              ; Branch NEQ error in msg
                05  12  OFE4  3166          BNEQ    30$
    57    0200 8F  A8  OFE6  3167 27$:     BISW2   #DLK$M_XMTACK,R7                ; Msg xmt'd ok
          63    04  91  OFEB  3168 30$:     CMPB    #DDCMP$C_STA_MNT,GF$B_STATE(R3) ; Do not set the bit is already
                05  13  OFEE  3169          BEQL    35$                             ; in maintenance mode
    57    0080 8F  A8  OFF0  3170          BISW    #DLK$M_MNTRCV,R7                ; Inform driver of maint mode
                05  OFF5  3171 35$:     RSB
                       OFF6  3172
                       OFF6  3173 ;DDCMP$CMPMNT - Complete the maintenance message
                       OFF6  3174 ;
                       OFF6  3175 ; This routine is called after the maint message has been sent to put
                       OFF6  3176 ; the message on the CMPQ for IO completion.
                       OFF6  3177 ;
                       OFF6  3178 ;       INPUTS  R5 = Address of the TF block
                       OFF6  3179 ;               R8 = Address of the XMT'd message
                       OFF6  3180 ;
```

```
                     0FF6  3181 ;       OUTPUTS R6,R7 contain return status for driver
                     0FF6  3182 ;               All other registers are preserved
                     0FF6  3183 DDCMP$CMPMNT:
      56    08   3C  0FF6  3184          MOVZWL  S^#DLK$C_RETINFO,R6             ; Set up return status
57    0400 8F   3C  0FF9  3185          MOVZWL  #DLK$M_XMTCMP,R7                ; for driver
  2C  B5    68    0E  0FFE  3186          INSQUE  (R8),@TF$Q_CMPQ+4(R5)           ; Put XMT in queue to comp
                   05  1002  3187          RSB
                       1003  3188
                       1003  3189
```

```
                        1003  3191          .SBTTL  Return error counters
                        1003  3192 ;DDCMP$RETERR - Return protocol error counters
                        1003  3193 ;
                        1003  3194 ; This routine is set up to return and/or clear error counters requested by
                        1003  3195 ; the calling program. It will return the parameters in the buffer specified
                        1003  3196 ; in R8, if there is no buffer then no parameters are returned.
                        1003  3197 ;
                        1003  3198 ;       INPUTS: R3 = GFB address
                        1003  3199 ;               R5 = TFB address
                        1003  3200 ;               R6 = Action asked of protocol
                        1003  3201 ;               R7 = Error counters to return
                        1003  3202 ;               R8 = Address of buffer to return in which to return counters
                        1003  3203 ;
                        1003  3204 ;       OUTPUTS R6,R7 contain the return status
                        1003  3205 ;               R8 contains the error counters
                        1003  3206 ;               The counters may or may not be cleared depending on the request
                        1003  3207 ;               R3 and R5 are preserved
                        1003  3208 ;
                        1003  3209 DDCMP$RETERR:
          1B 57   01    E1  1003  3210          BBC     #DLK$V_TRIB,R7,10$              ; If BC then assume GLOB cntrs
                  28    BB  1007  3211          PUSHR   #^M<R3,R5>                      ; Save registers
          56   1E A5    3C  1009  3212          MOVZWL  TF$W_TEB(R5),R6                 ; Get size to transfer
                  58    D5  100D  3213          TSTL    R8                              ; If EQL then no buffer
                  06    13  100F  3214          BEQL    5$                              ; specified
    68  018A C5    56    28  1011  3215          MOVC3   R6,TF$K_ERRSRT(R5),(R8)         ; Put counters into block
                  28    BA  1017  3216 5$:      POPR    #^M<R3,R5>                      ; Restore registers
          1D 57   00    E1  1019  3217          BBC     #DLK$V_CLEAR,R7,20$             ; If BC then do not clear cntrs
                01B2    30  101D  3218          BSBW    DDCMP$RESET_TCTRS               ; Else clear counters
                  18    11  1020  3219          BRB     20$
                  28    BB  1022  3220 10$:     PUSHR   #^M<R3,R5>                      ; Save registers
          56   0B A3    3C  1024  3221          MOVZWL  GF$W_GEB(R3),R6                 ; Get size to transfer
                  58    D5  1028  3222          TSTL    R8                              ; If EQL then no buffer
                  05    13  102A  3223          BEQL    15$                             ; specified
    68  1F A3    56    28  102C  3224          MOVC3   R6,GF$K_ERRSRT(R3),(R8)         ; Put counters into block
                  28    BA  1031  3225 15$:     POPR    #^M<R3,R5>                      ; Restore registers
          03 57   00    E1  1033  3226          BBC     #DLK$V_CLEAR,R7,20$             ; If BC then do not clear cntrs
                0188    30  1037  3227          BSBW    DDCMP$RESET_GCTRS               ; Else clear counters
          56    08    3C  103A  3228 20$:     MOVZWL  S^#DLK$C_RETINFO,R6             ; Set up return to driver
    57  2000 8F    A8  103D  3229          BISW2   #DLK$M_ERROR,R7                 ; A error block has been passed
                      1042  3230                                                  ; to driver
                  05  1042  3231          RSB
                      1043  3232
```

```
                        1043  3234
                        1043  3235          .SBTTL  Routines to handle sending and logging NAK's
                        1043  3236
                        1043  3237 ;DDCMP$LOGNAK
                        1043  3238 ;
                        1043  3239 ; This routine is called when a NAK is received to increment the proper
                        1043  3240 ; counter associated with the NAK subtype.
                        1043  3241 ;
                        1043  3242 ;       INPUTS          R3 = Address of the GF block
                        1043  3243 ;                       R5 = Address of the TF block
                        1043  3244 ;                       R8 = Address of teh received block
                        1043  3245 ;
                        1043  3246 ;       OUTPUTS         R2 = NAK type if NAK sent in reply to a REP
                        1043  3247 ;                       R3,R5,R8 are preserved
                        1043  3248 ;
                        1043  3249 ;
                        1043  3250 DDCMP$LOGNAK:
          00    EF      1043  3251          EXTZV   #MFD$C_SUBTYP,-         ; Get the NAK type
          06            1045  3252                  #MFD$C_SUBTYPL,-
    50  02  A8          1046  3253                  MFD$W_TYPFLG+1(R8),R0
    50  03  91          1049  3254          CMPB    S^#DDCMP$C_NAK3,R0              ; Branch EQL because XMT
          2B  13        104C  3255          BEQL    20$                             ; thresholds aren't counted
                        104E  3256                                                  ; on these NAK's
        0151  30        104E  3257          BSBW    DDCMP$INCR_XMT_THRS             ; Incr XMT threshold counter
    50  01  91          1051  3258          CMPB    S^#DDCMP$C_NAK7,R0              ; Branch NEQ not reason code 1
          0F  12        1054  3259          BNEQ    10$
51  01AA  C5  3E        1056  3260          MOVAW   TF$B_DEO(R5),R1                 ; Incr the data error outbound
        0078  30        105B  3261          BSBW    DDCMP$INCR_COUNTER              ; field
          01  A8        105E  3262          BISW2   #TF$M_OHCRC,-                   ; Set bit that NAK was set for
    01A8  C5            1060  3263                  TF$W_DEOBC(R5)                  ; a header CRC
          70  11        1063  3264          BRB     70$
    50  02  91          1065  3265 10$:     CMPB    S^#DDCMP$C_NAK2,R0              ; Branch NEQ not reason code 2
          21  12        1068  3266          BNEQ    30$
51  01AA  C5  3E        106A  3267          MOVAW   TF$B_DEO(R5),R1                 ; Incr the data error outbound
        0064  30        106F  3268          BSBW    DDCMP$INCR_COUNTER              ; field
          02  A8        1072  3269          BISW2   #TF$M_ODCRC,-                   ; Set bit that NAK was set for
    01A8  C5            1074  3270                  TF$W_DEOBC(R5)                  ; a data CRC
          5C  11        1077  3271          BRB     70$
51  01AA  C5  3E        1079  3272 20$:     MOVAW   TF$B_DEO(R5),R1                 ; Incr the data error outbound
        0055  30        107E  3273          BSBW    DDCMP$INCR_COUNTER              ; field
          04  A8        1081  3274          BISW2   #TF$M_OREP5,-                   ; Set bit that NAK was sent in
    01A8  C5            1083  3275                  TF$W_DEOBC(R5)                  ; reply to a REP
    52  50  D0          1086  3276          MOVL    R0,R2                           ; Set up R2 with NAK type
          4A  11        1089  3277          BRB     70$
    50  08  91          108B  3278 30$:     CMPB    S^#DDCMP$C_NAK8,R0              ; Branch NEQ not reason code 8
          0F  12        108E  3279          BNEQ    40$
51  01B9  C5  3E        1090  3280          MOVAW   TF$B_RBE(R5),R1                 ; Incr the remote buffer error
        003E  30        1095  3281          BSBW    DDCMP$INCR_COUNTER              ; field
          01  A8        1098  3282          BISW2   #TF$M_RBUF_NAVL,-               ; Set bit that NAK was sent
    01B7  C5            109A  3283                  TF$W_RBEBC(R5)                  ; because a RCV buffer was
                        109D  3284                                                  ; not available
          36  11        109D  3285          BRB     70$
    50  10  91          109F  3286 40$:     CMPB    S^#DDCMP$C_NAK16,R0             ; Branch NEQ not reason code 16
          0F  12        10A2  3287          BNEQ    50$
51  01B9  C5  3E        10A4  3288          MOVAW   TF$B_RBE(R5),R1                 ; Incr the remote buffer error
        002A  30        10A9  3289          BSBW    DDCMP$INCR_COUNTER              ; field
          02  A8        10AC  3290          BISW2   #TF$M_RBUF_SML,-                ; Set bit that NAK was sent
```

```
                01B7 C5          10AE  3291                    TF$W_RBEBC(R5)                  ; because a RCV buffer was
                                 10B1  3292                                                    ; too small
                     22    11    10B1  3293                BRB     70$
                50   09    91    10B3  3294 50$:           CMPB    S^#DDCMP$C_NAK9,R0          ; Branch NEQ not reason code 9
                      0D   12    10B6  3295                BNEQ    60$
              51   28 A3  9E    10B8  3296                MOVAB   GF$B_RSE(R3),R1             ; Incr the global remote
                   0017   30    10BC  3297                BSBW    DDCMP$INCR_COUNTER          ; station error counter
                      01  A8    10BF  3298                BISW2   #GF$M_RRCV_OVR,-            ; Set bit that NAK was sent
                   26 A3          10C1  3299                        GF$W_RSE_BCTRS(R3)          ; because of a RCV overrun
                      10   11    10C3  3300                BRB     70$
                50    11  91    10C5  3301 60$:           CMPB    S^#DDCMP$C_NAK17,R0         ; Branch NEQ not reason code 17
                      0B   12    10C8  3302                BNEQ    70$
              51   23 A3  9E    10CA  3303                MOVAB   GF$B_LSE(R3),R1             ; Incr the global local
                   0005   30    10CE  3304                BSBW    DDCMP$INCR_COUNTER          ; station error counter
                      08  A8    10D1  3305                BISW2   #GF$M_LMSGADR_FMT,-         ; Set bit that NAK was sent
                   21 A3          10D3  3306                        GF$W_LSE_BCTRS(R3)          ; because of a RCV overrun
                               05 10D5  3307 70$:           RSB
                                  10D6  3308
```

```
                        10D6  3310
                        10D6  3311 ;DDCMP$INCR_COUNTER - Increment the given counter by one
                        10D6  3312 ;
                        10D6  3313 ; This routine increments the byte counter pointed to by R1, but it will hold
                        10D6  3314 ; the counter at 255, instead of allowing the counter to wrap.
                        10D6  3315 ;
                        10D6  3316 ;       INPUTS:
                        10D6  3317 ;               R1 = Address of counter if increment
                        10D6  3318 ;
                        10D6  3319 ;       OUTPUTS:
                        10D6  3320 ;               R0 is destroyed
                        10D6  3321 ;
                        10D6  3322 DDCMP$INCR_COUNTER:
50   61   9A  10D6  3323          MOVZBL  (R1),R0                 ; Get current count
     50   96  10D9  3324          INCB    R0
     03   1D  10DB  3325          BVS     10$                     ; If VS then at limit
61   50   90  10DD  3326          MOVB    R0,(R1)                 ; Else increment counter
          05  10E0  3327 10$:     RSB
              10E1  3328
```

```
                     10E1  3330
                     10E1  3331 ;++
                     10E1  3332 ;DDCMP$SENDNAK
                     10E1  3333 ;
                     10E1  3334 ; This routine is called when the driver informs the protocol of some
                     10E1  3335 ; error it received when receiving the message. This routine covers the
                     10E1  3336 ; senting of the SNAK field for five NAK reasons (1,2,8,9,16) and also
                     10E1  3337 ; incrementing the counters associated with those reasons.
                     10E1  3338 ;
                     10E1  3339 ;       INPUT           R3 = Address of the GF block
                     10E1  3340 ;                       R5 = Address of the TF block
                     10E1  3341 ;                       R7 = The RCV message error from driver.
                     10E1  3342 ;
                     10E1  3343 ;       OUTPUTS         R3 and R5 are preserved
                     10E1  3344 ;                       R6 = Retrun information for driver
                     10E1  3345 ;                       R7 = Errors encountered
                     10E1  3346 ;
                     10E1  3347 DDCMP$SENDNAK:
    56   08   3C     10E1  3348          MOVZWL  S^#DLK$C_RETINFO,R6             ; Set up return status
         57   D4     10E4  3349          CLRL    R7                              ; for driver
                     10E6  3350
19  57   01   E0     10E6  3351          BBS     #DLK$V_HDRCRC,R7,10$
29  57   02   E0     10EA  3352          BBS     #DLK$V_DATACRC,R7,20$
39  57   03   E0     10EE  3353          BBS     #DLK$V_BUNAVL,R7,30$
49  57   04   E0     10F2  3354          BBS     #DLK$V_BUFTOSML,R7,40$
59  57   05   E0     10F6  3355          BBS     #DLK$V_RCVOVR,R7,50$
72  57   06   E0     10FA  3356          BBS     #DLK$V_PADBYTE,R7,60$
                     10FE  3357
    57   04   3C     10FE  3358          MOVZWL  #DLK$M_CMDERR,R7                ; Return error
         7E   11     1101  3359          BRB     70$
         01   90     1103  3360 10$:     MOVB    #DDCMP$C_NAK1,-                 ; Set the NAK reason to 1
    0188 C5          1105  3361                  TF$B_NAKRSN(R5)
51  01AF C5   3E     1108  3362          MOVAW   TF$B_DEI(R5),R1                 ; Incr the data error inbound
       FFC6   30     110D  3363          BSBW    DDCMP$INCR_COUNTER              ; field
         01   A8     1110  3364          BISW2   #TF$M_IHCRC,-                   ; Set bit that SNAK set because
    01AD C5          1112  3365                  TF$W_DEIBC(R5)                  ; a header err was encountered
         59   11     1115  3366          BRB     60$
                     1117  3367
         02   90     1117  3368 20$:     MOVB    #DDCMP$C_NAK2,-                 ; Set the NAK reason to 2
    0188 C5          1119  3369                  TF$B_NAKRSN(R5)
51  01AF C5   3E     111C  3370          MOVAW   TF$B_DEI(R5),R1                 ; Incr the data error inbound
       FFB2   30     1121  3371          BSBW    DDCMP$INCR_COUNTER              ; field
         02   A8     1124  3372          BISW2   #TF$M_IDCRC,-                   ; Set bit that SNAK set because
    01AD C5          1126  3373                  TF$W_DEIBC(R5)                  ; a data err was encountered
         45   11     1129  3374          BRB     60$
         08   90     112B  3375 30$:     MOVB    #DDCMP$C_NAK8,-                 ; Set the NAK reason to 8
    0188 C5          112D  3376                  TF$B_NAKRSN(R5)
51  01B4 C5   3E     1130  3377          MOVAW   TF$B_LBE(R5),R1                 ; Incr the local buffer error
       FF9E   30     1135  3378          BSBW    DDCMP$INCR_COUNTER              ; field
         01   A8     1138  3379          BISW2   #TF$M_LBUF_NAVL,-               ; Set bit that SNAK set because
    01B2 C5          113A  3380                  TF$W_LBEBC(R5)                  ; the local buffer was unavl
         31   11     113D  3381          BRB     60$
         10   90     113F  3382 40$:     MOVB    #DDCMP$C_NAK16,-                ; Set the NAK reason to 16
    0188 C5          1141  3383                  TF$B_NAKRSN(R5)
51  01B4 C5   3E     1144  3384          MOVAW   TF$B_LBE(R5),R1                 ; Incr the local buffer error
       FF8A   30     1149  3385          BSBW    DDCMP$INCR_COUNTER              ; field
         02   A8     114C  3386          BISW2   #TF$M_LBUF_SML,-                ; Set bit that SNAK set because
```

```
                01B2 C5          114E  3387                          TF$W_LBEBC(R5)                  ; the local buffer was too sml
                     1D    11    1151  3388                  BRB     60$
                     09    90    1153  3389 50$:             MOVB    #DDCMP$C_NAK9,-                 ; Set NAK reason code 9
                0188 C5          1155  3390                          TF$B_NAKRSN(R5)
5:                23 A3    3E    1158  3391                  MOVAW   GF$B_LSE(R3),R1                 ; Incr global error local
                   FF77    30    115C  3392                  BSBW    DDCMP$INCR_COUNTER              ; station error
                63   01    91    115F  3393                  CMPB    #DDCMP$C_STA_RUN,GF$B_STATE(R3) ; Branch NEQ the SNAK is set -
                     06    12    1162  3394                  BNEQ    55$                             ; only in RUN state
                     01    A8    1164  3395                  BISW2   #GF$M_LRCV_OVR,-                ; Set the reason for the SNAK
                   21 A3          1166  3396                          GF$W_LSE_BCTRS(R3)              ; as local RCV overrun
                     06    11    1168  3397                  BRB     60$
                     02    A8    116A  3398 55$:             BISW2   #GF$M_LNRCV_OVR,-               ; Set the that the NAK is
                   21 A3          116C  3399                          GF$W_LSE_BCTRS(R3)              ; not sent
                     0F    11    116E  3400                  BRB     65$
                54   02    9A    1170  3401 60$:             MOVZBL  S^#DDCMP$C_NAK,R4               ; Set up to queue NAK
                   FB13    30    1173  3402                  BSBW    DDCMP$QUEUE_XMT                 ; Queue the NAK
                     57    95    1176  3403                  TSTB    R7                              ; Branch on EQL all is well
                     05    13    1178  3404                  BEQL    65$
                     01    88    117A  3405                  BISB2   #TF$M_SNAK,-                    ; Else set that a NAK needs to
                0189 C5          117C  3406                          TF$B_TFLAGS(R5)                 ; be sent
                     01    10    117F  3407 65$:             BSBB    DDCMP$INCR_RCV_THRS             ; Incr the RCV threshold cntrs
                           05    1181  3408 70$:             RSB
                                 1182  3409
```

```
                        1182  3411
                        1182  3412          .SBTTL  Routines to increment threshold counter
                        1182  3413
                        1182  3414 ;DDCMP$INCR_RCV_THRS, DDCMP$INCR_XMT_THRS
                        1182  3415 ;
                        1182  3416 ; These routines increment the receive or transmit threshold error fields.
                        1182  3417 ; If a threshold has been exceeded the routine reports the error and
                        1182  3418 ; clears the field.
                        1182  3419 ;
                        1182  3420 ;       INPUTS          R3 = Address of the GF block
                        1182  3421 ;                       R5 = Address of the TF block
                        1182  3422 ;
                        1182  3423 ;       OUTPUTS         R3 and R5 are preserved
                        1182  3424 ;                       R7 will have a bit set if the threshold has
                        1182  3425 ;                       been exceeded
                        1182  3426 ;
                        1182  3427 DDCMP$INCR_RCV_THRS:
                  51 DD 1182  3428          PUSHL   R1
                  03 EF 1184  3429          EXTZV   #DDCMP$C_RCV_THRS,-             ; Extract the field
                  03    1186  3430                  #DDCMP$C_THR_SIZE,-             ; to increment
           51 01C5 C5   1187  3431                  TF$W_THRES(R5),R1
                  51 96 118B  3432          INCB    R1                              ; Incr the thres error counter
               51 07 91 118D  3433          CMPB    #DDCMP$C_THR_MAX,R1             ; Has the thres been exceeded
                  05 14 1190  3434          BGTR    10$                             ; No
               57 01 A8 1192  3435          BISW2   #DLK$M_TRNSERR,R7               ; Report that thres exceeded
                  51 D4 1195  3436          CLRL    R1                              ; Set the RCV threshold count
                        1197  3437                                                  ; to zero
                        1197  3438
            03    51 F0 1197  3439 10$:     INSV    R1,#DDCMP$C_RCV_THRS,-          ; Insert count
                  03    119A  3440                  #DDCMP$C_THR_SIZE,-
            01C5  C5    119B  3441                  TF$W_THRES(R5)
                  51 8ED0 119E  3442          POPL    R1
                     05 11A1  3443          RSB
                        11A2  3444
                        11A2  3445 DDCMP$INCR_XMT_THRS:
                  51 DD 11A2  3446          PUSHL   R1
                  00 EF 11A4  3447          EXTZV   #DDCMP$C_XMT_THRS,-             ; Extract the field
                  03    11A6  3448                  #DDCMP$C_THR_SIZE,-             ; to increment
           51 01C5 C5   11A7  3449                  TF$W_THRES(R5),R1
                  51 96 11AB  3450          INCB    R1                              ; Incr the thres error counter
               51 07 91 11AD  3451          CMPB    #DDCMP$C_THR_MAX,R1             ; Has the thres been exceeded
                  05 14 11B0  3452          BGTR    10$                             ; No
               57 01 A8 11B2  3453          BISW2   #DLK$M_TRNSERR,R7               ; Report that thres exceeded
                  51 D4 11B5  3454          CLRL    R1                              ; Set the XMT threshold count
                        11B7  3455                                                  ; to zero
                        11B7  3456
            00    51 F0 11B7  3457 10$:     INSV    R1,#DDCMP$C_XMT_THRS,-          ; Insert count
                  03    11BA  3458                  #DDCMP$C_THR_SIZE,-
            01C5  C5    11BB  3459                  TF$W_THRES(R5)
                  51 8ED0 11BE  3460          POPL    R1
                     05 11C1  3461          RSB
                        11C2  3462
```

```
                        11C2  3464
                        11C2  3465          .SBTTL  Read or clear error counters
                        11C2  3466
                        11C2  3467 ; RESET_GCTRS - Clear global counters
                        11C2  3468 ;
                        11C2  3469 ; This routine zeros the global counter fields.
                        11C2  3470 ;
                        11C2  3471 ;       INPUTS          R3 = Address of GF block
                        11C2  3472 ;                       R5 = Address of TF block
                        11C2  3473 ;
                        11C2  3474 ;       OUTPUTS         R3 and R5 are preserved
                        11C2  3475 ;                       Global counters are zeroed
                        11C2  3476 ;
                        11C2  3477 DDCMP$RESET_GCTRS:
                        11C2  3478
21 A3    B4             11C2  3479          CLRW    GF$W_LSE_BCTRS(R3)
23 A3    94             11C5  3480          CLRB    GF$B_LSE(R3)
26 A3    B4             11C8  3481          CLRW    GF$W_RSE_BCTRS(R3)
28 A3    94             11CB  3482          CLRB    GF$B_RSE(R3)
                        11CE  3483
                        11CE  3484          ASSUME  GF$B_MDF_CRC EQ GF$B_GH_CRC+1
                        11CE  3485
29 A3    B4             11CE  3486          CLRW    GF$B_GH_CRC(R3)
                        11D1  3487
         05             11D1  3488          RSB
                        11D2  3489
                        11D2  3490 ; RESET_TCTRS - Clear tributary error counters
                        11D2  3491 ;
                        11D2  3492 ; This routine clear error counters kept for specfic tributaries. Clearing
                        11D2  3493 ; them for one tributary will not nessicarily clear them for all tributaries.
                        11D2  3494 ;
                        11D2  3495 ;       INPUT           R5 = Adress of the TF block
                        11D2  3496 ;
                        11D2  3497 ;       OUTPUT          Tributary coutners are cleared
                        11D2  3498 ;
                        11D2  3499 DDCMP$RESET_TCTRS:
01A8 C5  B4             11D2  3500          CLRW    TF$W_DEOBC(R5)
01AA C5  94             11D6  3501          CLRB    TF$B_DEO(R5)
01AD C5  B4             11DA  3502          CLRW    TF$W_DEIBC(R5)
01AF C5  94             11DE  3503          CLRB    TF$B_DEI(R5)
01C1 C5  94             11E2  3504          CLRB    TF$B_LRTO(R5)
01C4 C5  94             11E6  3505          CLRB    TF$B_RRTO(R5)
01BC C5  B4             11EA  3506          CLRW    TF$W_STOBC(R5)
01BE C5  94             11EE  3507          CLRB    TF$B_STO(R5)
01B2 C5  B4             11F2  3508          CLRW    TF$W_LBEBC(R5)
01B4 C5  94             11F6  3509          CLRB    TF$B_LBE(R5)
01B7 C5  B4             11FA  3510          CLRW    TF$W_RBEBC(R5)
01B9 C5  94             11FE  3511          CLRB    TF$B_RBE(R5)
01A4 C5  B4             1202  3512          CLRW    TF$W_SELSP(R5)
019E C5  D4             1206  3513          CLRL    TF$L_DMSGX(R5)
0198 C5  D4             120A  3514          CLRL    TF$L_DMSGR(R5)
0192 C5  D4             120E  3515          CLRL    TF$L_DBYTX(R5)
018C C5  D4             1212  3516          CLRL    TF$L_DBYTR(R5)
                        1216  3517
01C5 C5  B4             1216  3518          CLRW    TF$W_THRES(R5)
                        121A  3519
         05             121A  3520          RSB
```

```
                121B  3521
 00             121B  3522                HALT                            ; Problem if it gets here
                121C  3523 DRIVER_END::
                121C  3524                .END
```

```
$$$TYP                  = 000003E8
$$$WID                  = 00006000
$$BASE                  = 00000001
$$DISPL                 = 00000005
$$GENSW                 = 00000001
$$HIGH                  = 00000004
$$LIMIT                 = 00000003
$$LOW                   = 00000001
$$MNSW                  = 00000001
$$MXSW                  = 00000001
ACT$_BUG                = 00000000
ACT$_CMPMNT             = 0000000A
ACT$_HALT               = 00000004
ACT$_IGNORE             = 00000001
ACT$_MAINT              = 00000006
ACT$_QEMPTY             = 0000000C
ACT$_RCVMSG             = 00000002
ACT$_REPTIM             = 00000009
ACT$_RETERR             = 0000000E
ACT$_SENDNAK            = 00000010
ACT$_SENDSTRT           = 00000003
ACT$_SETCHAR            = 0000000F
ACT$_STARTACK           = 00000005
ACT$_STARTING           = 00000008
ACT$_STARTINT           = 0000000B
ACT$_STARTMNT           = 0000000D
ACT$_START_TIMER        = 00000011
ACT$_STOP_TIMER         = 00000012
ACT$_XMTMSG             = 00000007
BUG$_NOBUFPCKT            ********    X   01
CHANGE_STA                00000153 R      01
CRCTABLE                  00000084 R      01
DDCMP                     000000C4 RG     01
DDCMP$AB_STTAB            00000000 R      01
DDCMP$BUG                 0000025E R      01
DDCMP$CHK_HDR             00000821 R      01
DDCMP$CMPANT              00000FF6 R      01
DDCMP$C_ACK             = 00000001
DDCMP$C_ACTBITS         = 00000005
DDCMP$C_DEL             = 000000FF
DDCMP$C_DLE             = 00000090
DDCMP$C_ENQ             = 00000005
DDCMP$C_EOINTVL         = 00000002
DDCMP$C_EV_CHR          = 0000000B
DDCMP$C_EV_ERR          = 0000000A
DDCMP$C_EV_LER          = 0000000D
DDCMP$C_EV_LNK          = 0000000C
DDCMP$C_EV_MNT          = 00000009
DDCMP$C_EV_QMT          = 00000008
DDCMP$C_EV_RCV          = 00000000
DDCMP$C_EV_REP          = 00000005
DDCMP$C_EV_RMM          = 00000003
DDCMP$C_EV_SPT          = 0000000F
DDCMP$C_EV_SRT          = 00000006
DDCMP$C_EV_STA          = 00000002
DDCMP$C_EV_STI          = 00000001
DDCMP$C_EV_STP          = 00000007
DDCMP$C_EV_STT          = 0000000E
DDCMP$C_EV_XMT          = 00000004
DDCMP$C_FICL            = 00000000
DDCMP$C_HEADER          = 00000006
DDCMP$C_MAX_EV          = 0000000F
DDCMP$C_MILCI           = 000003E8
DDCMP$C_MODULO          = 0000007F
DDCMP$C_MSGCNT_LIMIT    = 00000007
DDCMP$C_NAK             = 00000002
DDCMP$C_NAK1            = 00000001
DDCMP$C_NAK16           = 00000010
DDCMP$C_NAK17           = 00000011
DDCMP$C_NAK2            = 00000002
DDCMP$C_NAK3            = 00000003
DDCMP$C_NAK8            = 00000008
DDCMP$C_NAK9            = 00000009
DDCMP$C_NUMEVT          = 00000010
DDCMP$C_NUMSTA          = 00000008
DDCMP$C_NXTINTVL        = 00000003
DDCMP$C_QUA             = 000000FA
DDCMP$C_RCV_THRS        = 00000003
DDCMP$C_REACLOCK        = 00000001
DDCMP$C_REP             = 00000003
DDCMP$C_RUNTQE          = 004C4B40
DDCMP$C_SEL_THRS        = 00000006
DDCMP$C_SIZEOFQ         = 0000007F
DDCMP$C_SOH             = 00000081
DDCMP$C_STABITS         = 00000003
DDCMP$C_STACK           = 00000007
DDCMP$C_STAMSK          = 000000E0
DDCMP$C_STA_AST         = 00000003
DDCMP$C_STA_FR1         = 00000005
DDCMP$C_STA_FR2         = 00000006
DDCMP$C_STA_FR3         = 00000007
DDCMP$C_STA_HLT         = 00000000
DDCMP$C_STA_IST         = 00000002
DDCMP$C_STA_MNT         = 00000004
DDCMP$C_STA_RUN         = 00000001
DDCMP$C_STPTQE          = 01C9C380
DDCMP$C_STRT            = 00000006
DDCMP$C_SYN             = 00000096
DDCMP$C_THR_MAX         = 00000007
DDCMP$C_THR_SIZE        = 00000003
DDCMP$C_TRIBTQE         = 002625A0
DDCMP$C_XMT_THRS        = 00000000
DDCMP$ENDTABLES         = 00000080 R      01
DDCMP$ERROR               0000022E R      01
DDCMP$EVENT               000001A3 R      01
DDCMP$HALT                00000F1C R      01
DDCMP$IGNORE              0000023E R      01
DDCMP$INCR_COUNTER        000010D6 R      01
DDCMP$INCR_RCV_THRS       00001182 R      01
DDCMP$INCR_XMT_THRS       000011A2 R      01
DDCMP$INS_RTOQ            00000E6F R      01
DDCMP$LOGNAK              00001043 R      01
DDCMP$MAINT               00000FAD R      01
DDCMP$MAINT_HALT          00000F43 R      01
```

```
DDCMP$M_BIT16                   = 00010000
DDCMP$M_BIT24                   = 01000000
DDCMP$M_BIT8                    = 00000100
DDCMP$QEMPTY                      000008AE R      01
DDCMP$QEMPTY_ALT                  000008B9 R      01
DDCMP$QUEUE_XMT                   00000989 R      01
DDCMP$RCVMSG                      00000564 R      01
DDCMP$REM_RTOQ                    00000EBC R      01
DDCMP$REPORT                      00000265 R      01
DDCMP$REPTIM                      00000893 R      01
DDCMP$REP_TO                      00000DED R      01
DDCMP$REQQUEUE_XMT                00000AD6 R      01
DDCMP$RESET_GCTRS                 000011C2 R      01
DDCMP$RESET_TCTRS                 000011D2 R      01
DDCMP$RESET_VAR                   00000439 R      01
DDCMP$RETERR                      00001003 R      01
DDCMP$SELECT                      00000D45 R      01
DDCMP$SENDNAK                     000010E1 R      01
DDCMP$SENDSTRT                    000003DA R      01
DDCMP$SETCHAR                     00000269 R      01
DDCMP$SETCRC                      00000BA9 R      01
DDCMP$SETUP_ENQ_HDR               00000C23 R      01
DDCMP$SETUP_MSG_HDR               00000BD2 R      01
DDCMP$STARTACK                    00000517 R      01
DDCMP$STARTING                    00000245 R      01
DDCMP$STARTINT                    000003E1 R      01
DDCMP$STARTMNT                    00000548 R      01
DDCMP$START_SEL_TIMER             000008C7 R      01
DDCMP$START_TIMER                 000008D8 R      01
DDCMP$STOP_SEL_TIMER              00000985 R      01
DDCMP$STOP_TIMER                  0000095B RG     01
DDCMP$TIMER                       00000CE1 RG     01
DDCMP$TIMER_END                   00000E57 R      01
DDCMP$V_BIT6                    = 00000010
DDCMP$V_BIT24                   = 00000018
DDCMP$V_BIT8                    = 00000008
DDCMP$XMTERR                      00000810 R      01
DDCMP$XMTMSG                      00000704 R      01
DDCMP$XMTMSG_ALT                  00000724 R      01
DLA$A_POSTQ                       00000004
DLA$A_XMT_INPR                    00000000
DLA$C_ADDR_LENGTH                 00000008
DLA$K_ADDR_LENGTH                 00000008
DLK$B_MAINT                       00000008
DLK$B_MAXREP                      00000002
DLK$B_MAXSEL                      00000003
DLK$B_MRB                         00000009
DLK$B_MSGCNT                      00000001
DLK$B_TRIB                        00000000
DLK$C_ACTNOTCOM                 = 00000009
DLK$C_CHAR                      = 00000005
DLK$C_RCVMSG                    = 00000001
DLK$C_REQEBA                    = 00000003
DLK$C_RETINFO                   = 00000008
DLK$C_START_TIMER               = 00000006
DLK$C_STOP_TIMER                = 00000007
DLK$C_USRINT                    = 00000004
DLK$C_XMTMSG                    = 00000002
DLK$M_BADACTION                 = 00000002
DLK$M_BADSTATE                  = 00000001
DLK$M_CMDERR                    = 00000004
DLK$M_CRC                       = 00010000
DLK$M_ERROR                     = 00002000
DLK$M_HDRERR                    = 00000008
DLK$M_MNTRCV                    = 00000080
DLK$M_MSGSENT                   = 00000001
DLK$M_PRSTERR                   = 00000002
DLK$M_QFULERR                   = 00000020
DLK$M_RCVACK                    = 00000100
DLK$M_RCVERR                    = 00000008
DLK$M_REINT                     = 00000004
DLK$M_STRTRCV                   = 00000040
DLK$M_TMREXPD                   = 00008000
DLK$M_TRNLK                     = 00004000
DLK$M_TRNSERR                   = 00000001
DLK$M_XMTACK                    = 00000200
DLK$M_XMTCMP                    = 00000400
DLK$M_XMTERR                    = 00000010
DLK$V_BABBLE                    = 0000000A
DLK$V_BUFTOSML                  = 00000004
DLK$V_BUNAVL                    = 00000003
DLK$V_CLEAR                     = 00000000
DLK$V_CRC                       = 00000010
DLK$V_DATACRC                   = 00000002
DLK$V_DUPLEX                    = 00000002
DLK$V_HDRCRC                    = 00000001
DLK$V_LNKLOST                   = 00000000
DLK$V_LNTYP                     = 00000003
DLK$V_MAINT                     = 00000002
DLK$V_MSGCNT                    = 00000004
DLK$V_MSGSENT                   = 00000000
DLK$V_PADBYTE                   = 00000006
DLK$V_POINT                     = 00000001
DLK$V_QEMPTY                    = 00000001
DLK$V_QFULERR                   = 00000005
DLK$V_RCVOVR                    = 00000005
DLK$V_REPTIM                    = 00000007
DLK$V_REPWAIT                   = 00000008
DLK$V_SELTIM                    = 00000005
DLK$V_SELWAIT                   = 00000006
DLK$V_SETDEF                    = 0000000B
DLK$V_START                     = 00000000
DLK$V_STATYP                    = 00000000
DLK$V_STOP                      = 00000001
DLK$V_STREAM                    = 00000009
DLK$V_TRIB                      = 00000001
DLK$W_REPWAIT                     00000004
DLK$W_SELWAIT                     00000006
DMP$CHAR                          000000FA R      01
DMP$RCVMSG                        0000013A R      01
DMP$REQEBA                        00000100 R      01
DMP$STPTMER                       00000134 R      01
DMP$STRTMER                       0000012E R      01
DMP$USRINT                        00000106 R      01
```

```
DMP$XMTMSG                      000000E2 R      01
DRIVER_END                      0000121C RG     01
DYN$C_BUFIO                   = 00000013
EXE$AC_TQENOREPT                ******** X      01
EXE$GL_ABSTIM                   ***** **  X     01
EXE$GQ_SYSTIME                  ****.***  X     01
EXE$INSTIMQ                     ******** X      01
EXE$RMVTIMQ                     ******** X      01
GF$B_BABTIM                     00000016
GF$B_CURSEL                     00000008
GF$B_DIPL                       0000000A
GF$B_DRVCHR                     00000002
GF$B_FIPL                       00000009
GF$B_GM_CRC                     00000029
GF$B_LSE                        00000023
GF$B_MAXSEL                     00000007
GF$B_MDF_CRC                    0000002A
GF$B_RSE                        00000028
GF$B_STATE                      00000000
GF$B_STRTIM                     00000015
GF$B_STRTSEL                    0000002B
GF$B_TIMER_STATE                00000001
GF$C_ERREND                     00000029
GF$C_ERRSRT                     0000001F
GF$C_LENGTH                     0000002C
GF$K_ERREND                     00000029
GF$K_ERRSRT                     0000001F
GF$K_LENGTH                     0000002C
GF$L_BABTMR                     0000001B
GF$L_SELEND                     0000000D
GF$L_STRTMR                     00000017
GF$L_TQE_STS                    00000011
GF$M_CRC                      = 00000001
GF$M_LMSGHDR_FMT              = 00000008
GF$M_LNRCV_OVR                = 00000002
GF$M_LRCV_OVR                 = 00000001
GF$M_LXMT_UNDER               = 00000004
GF$M_RRCV_OVR                 = 00000001
GF$M_TIMER_RUNNING            = 00000001
GF$V_CRC                      = 00000000
GF$V_TIMER_RUNNING            = 00000000
GF$W_GEB                        0000000B
GF$W_LSETYP                     0000001F
GF$W_LSE_BCTRS                  00000021
GF$W_NEXTCNT                    00000005
GF$W_RSETYP                     00000024
GF$W_RSE_BCTRS                  00000026
GF$W_SELOAI                     00000003
MFD$B_ADDR                      00000005
MFD$B_MSGID                     00000000
MFD$B_NUMB                      00000004
MFD$B_RESP                      00000003
MFD$C_LENGTH                    00000006
MFD$C_SUBTYP                  = 00000000
MFD$C_SUBTYPL                 = 00000006
MFD$K_LENGTH                    00000006
MFD$M_QSYNC                   = 00004000
MFD$M_SELECT                  = 00008000
MFD$V_SELECT                  = 0000000F
MFD$W_CNTFLG                    00000001
MFD$W_TYPFLG                    00000001
NMA$C_CTCIR_BRC               = 000003E8
NMA$C_CTCIR_BSN               = 000003E9
NMA$C_CTCIR_DBR               = 000003F2
NMA$C_CTCIR_DBS               = 000003F3
NMA$C_CTCIR_DEI               = 000003FC
NMA$C_CTCIR_DEO               = 000003FD
NMA$C_CTCIR_LBE               = 00000411
NMA$C_CTCIR_LRT               = 00000407
NMA$C_CTCIR_RBE               = 00000410
NMA$C_CTCIR_RRT               = 00000406
NMA$C_CTCIR_SIE               = 0000041A
NMA$C_CTCIR_SLT               = 0000041B
NMA$C_CTLIN_LPE               = 0000044D
NMA$C_CTLIN_RPE               = 0000044C
NMA$M_CNT_COU                 = 00008000
NMA$M_CNT_MAP                 = 00001000
NMA$M_CNT_TYP                 = 00000FFF
NMA$V_CNT_WID                 = 0000000D
POLY                            00000080 R      01
PR$_IPL                         ********  X     01
SET$DUPLEX                      00000362 R      01
SET$LNTYP                       00000370 R      01
SET$MSGCNT                      00000375 R      01
SET$POINT                       00000359 R      01
SET$REPTIM                      000003BB R      01
SET$REPWAIT                     000003C2 R      01
SET$SELTIM                      0000037B R      01
SET$SELWAIT                     00000388 R      01
SET$STATYP                      00000344 R      01
TF$A_BUFPTR                     00000048
TF$A_DEV_TIMER                  00000050
TF$A_GFB                        0000004C
TF$A_POSTQ                      00000058
TF$A_XMT_INPR                   00000054
TF$B_A                          00000013
TF$B_ADDR                       0000000D
TF$B_CHAR                       0000000C
TF$B_CURRTO                     0000001B
TF$B_DEI                        000001AF
TF$B_DEO                        000001AA
TF$B_LBE                        000001B4
TF$B_LRTO                       000001C1
TF$B_MAXRTO                     0000001A
TF$B_MMCTR                      0000001D
TF$B_MSGCNT                     0000001C
TF$B_N                          00000012
TF$B_NAKRSN                     00000188
TF$B_R                          00000011
TF$B_RADDR                      0000000F
TF$B_RBE                        000001B9
TF$B_REPTIM                     00000017
TF$B_RRTO                       000001C4
TF$B_SELTIM                     00000016
```

```
TF$B_SFLAGS             00000189      TF$V_STATYP             =  00000000
TF$B_STO                000001BE      TF$V_TIMER              =  00000002
TF$B_T                  00000014      TF$W_DBRTYP                00000188
TF$B_X                  00000015      TF$W_DBXTYP                00000190
TF$B_XADDR              0000000E      TF$W_DEIBC                 000001AD
TF$B_XQCNT              00000010      TF$W_DEITYP                000001AB
TF$C_ERREND             000001C5      TF$W_DEOBC                 000001A8
TF$C_ERRSTRT            0000018A      TF$W_DEOTYP                000001A6
TF$C_LENGTH             000001C8      TF$W_DMRTYP                00000196
TF$K_ERREND             000001C5      TF$W_DMXTYP                0000019C
TF$K_ERRSTRT            0000018A      TF$W_LBEBC                 000001B2
TF$K_LENGTH             000001C8      TF$W_LBETYP                000001B0
TF$L_DBYTR              0000018C      TF$W_LRTOTYP               000001BF
TF$L_DBYTX              00000192      TF$W_RBEBC                 000001B7
TF$L_DMSGR              00000198      TF$W_RBETYP                000001B5
TF$L_DMSGX              0000019E      TF$W_REPWAI                00000018
TF$L_HDR                00000000      TF$W_RRTOTYP               000001C2
TF$L_QACK               0000005C      TF$W_SELSP                 000001A4
TF$L_QNAK               0000008C      TF$W_SELTYP                000001A2
TF$L_QREP               000000BC      TF$W_STOBC                 000001BC
TF$L_QSTACK             0000011C      TF$W_STOTYP                000001BA
TF$L_QSTRT              000000EC      TF$W_TEB                   0000001E
TF$L_TQE                0000014C      TF$W_THRES                 000001C5
TF$M_DUPLEX           = 00000004      TQE$B_RMOD              =  00000028
TF$M_IDCRC            = 00000002      TQE$B_RQTYPE            =  0000000B
TF$M_IHCRC            = 00000001      TQE$C_SSREPT            =  00000005
TF$M_INCREP_SEL       = 00000002      TQE$C_SSSNGL            =  00000001
TF$M_IREPS            = 00000004      TQE$L_FPC               =  0000000C
TF$M_LBUF_NAVL        = 00000001      TQE$L_FR3               =  00000010
TF$M_LBUF_SML         = 00000002      TQE$L_FR4               =  00000014
TF$M_NOREP_SEL        = 00000001      TQE$Q_DELTA             =  00000020
TF$M_ODCRC            = 00000002      XMTQ$B_BUFTYP              0000000A
TF$M_OHCRC            = 00000001      XMTQ$B_FLAG                0000001F
TF$M_OREPS            = 00000004      XMTQ$B_MSGHDR              00000024
TF$M_OWNSELT          = 00000002      XMTQ$B_SLOT                0000001C
TF$M_POINT            = 00000002      XMTQ$C_LENGTH              0000002A
TF$M_RBUF_NAVL        = 00000001      XMTQ$K_LENGTH              0000002A
TF$M_RBUF_SML         = 00000002      XMTQ$L_BACC                00000014
TF$M_RCVDET           = 00000008      XMTQ$L_IRP                 0000000C
TF$M_SACK             = 00000002      XMTQ$L_TIMEND              00000010
TF$M_SELECT           = 00000001      XMTQ$M_CONTROL          =  00000004
TF$M_SNAK             = 00000001      XMTQ$M_SELECT           =  00000002
TF$M_SREP             = 00000004      XMTQ$Q_LINK                00000000
TF$M_STATYP           = 00000001      XMTQ$V_CONTROL          =  00000002
TF$M_TIMER            = 00000004      XMTQ$V_ONQUEUE          =  00000000
TF$Q_CMPQ                 00000028    XMTQ$W_BUFLEN              00000008
TF$Q_CTLQ                 00000020    XMTQ$W_DCRC                00000022
TF$Q_RTOQ                 00000038    XMTQ$W_ERROR               0000001D
TF$Q_XMTOVF               00000040    XMTQ$W_HCRC                00000020
TF$Q_XMTQ                 00000030    XMTQ$W_MSGOFF              00000018
TF$V_OWNSELT          = 00000001      XMTQ$W_MSGSIZE             0000001A
TF$V_POINT            = 00000001      _$ACT_NDX               =  00000013
TF$V_RCVDET           = 00000003      _$EVCOD                 =  0000000F
TF$V_SACK             = 00000001      _$PHASE                 =  00000000
TF$V_SELECT           = 00000000
TF$V_SNAK             = 00000000
TF$V_SREP             = 00000002
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	0000121C (4636.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ABS	000001C8 (456.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.06	00:00:01.35
Command processing	145	00:00:00.50	00:00:03.63
Pass 1	910	00:00:20.33	00:01:19.04
Symbol table sort	0	00:00:01.94	00:00:05.42
Pass 2	473	00:00:06.05	00:00:24.07
Symbol table output	2	00:00:00.25	00:00:00.69
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1570	00:00:29.15	00:01:54.22

The working set limit was 2400 pages.
164178 bytes (321 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1593 non-local and 361 local symbols.
3524 source lines were read in Pass 1, producing 29 object records in Pass 2.
45 pages of virtual memory were used to define 42 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[DRIVER.OBJ]SYNCHLIB.MLB;1	7
_255DUA28:[SHRLIB]NMALIBRY.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	8
_255DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	20

1659 GETS were required to define 20 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:DDCMP/OBJ=OBJ$:DDCMP MSRC$:DDCMP/UPDATE=(ENH$:DDCMP)+EXECML$/LIB+SHRLIB$:NMALIBRY/LIB+LIB$:SYNCHLIB/LIB

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DBDRIVER
LIS

CRDRIVER
LIS

CVDRIVER
LIS

DDDRIVER
LIS

DDDMP
LIS